VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

# Vícefaktorová autentizace pomocí protokolu Radius a biometrie v IPv6 sítích

# Multi-factor Authentication with Radius and Biometric in IPv6 Networks

2016 Bc. Dominik Michalina

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

# Zadání diplomové práce

Student: **Bc. Dominik Michalina**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2601T013 Telekomunikační technika

Téma: Vícefaktorová autentizace pomocí protokolu Radius a biometrie v IPv6 sítích.
Multi-factor Authentication with Radius and Biometric in IPv6 Networks

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem diplomové práce je navrhnout řešení s vícefaktorovou autentizací pomocí protokolu Radius a biometrie v IPv6 sítích.

1. Studium a popis autentizačních metod.
2. Studium a popis biometrických metod.
3. Návrh řešení autentizace v IPv6 sítích na bázi biometrie.
4. Ověření navrženého řešení v laboratorních podmínkách.

Seznam doporučené odborné literatury:

[1] Dirk van der Walt FreeRADIUS Beginner's Guide Packt Publishing 2011.
ISBN 978-1849514088

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Pavel Nevlud**

Datum zadání: 01.09.2015

Datum odevzdání: 29.04.2016

VYSOKÁ ŠKOLA BÁŇSKÁ - TECHNICKÁ UNIVERZITA * OSTRAVA * Fakulta elektrotechniky a informatiky

doc. Ing. Miroslav Vozňák, Ph.D.
*vedoucí katedry*

prof. RNDr. Václav Snášel, CSc.
*děkan fakulty*

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. dubna 2016 .....

Rád by som poďakoval môjmu vedúcemu diplomovej práce pánovi Ing. Pavlovi Nevludovi za odbornú pomoc, cenné rady a konzultáciu pri vytváraní tejto práce.

**Abstrakt**

Táto práca je venovaná návrhu autentifikačného systému s využitím protokolu RADIUS a biometrie v IPv6 sieťach. Návrh prebiehal na operačnom systéme Ubuntu a na implementáciu jednotlivých autentifikačných mechanizmov boli využité PAM moduly. Teoretická časť práce podrobne popisuje autentifikačné metódy, pričom je kladený dôraz na biometrickú autentifikáciu. V praktickej časti je detailne popísaná konfigurácia navrhnutého autentifikačného systému v danom OS. Ako biometrický prvok je použitá čítačka odtlačkov prstov. Navrhnutý systém bol napokon overený v laboratórnych podmienkach.

**Kľúčové slová**: RADIUS, FreeRADIUS, autentifikácia, heslo, token, biometria, PAM, IPv6, DNS

**Abstract**

This work is dedicated to the design of an authentication system using RADIUS protocol and biometrics in IPv6 networks. The system design took place on Ubuntu operating system and PAM modules were used to implement each one of the authentication mechanisms. The theoretical part carefully describes authentication methods accentuating biometrics authentication. The practical part describes configuration of the suggested authentication system in given OS. Finger prints scanner is used as a mean of biometrics. At last, the suggested system was verified in laboratory conditions.

**Key Words**: RADIUS, FreeRADIUS, authentication, password, token, biometric, PAM, IPv6, DNS

# Obsah

# Zoznam použitých skratiek a symbolov

| | | |
|---|---|---|
| 2D | – | Two-Dimensional |
| 3D | – | Three-Dimensional |
| A | – | Address |
| AAA | – | Authentication, Authorization and Accounting |
| AAAA | – | Quad-A |
| AVP | – | Attribute Value Pairs |
| CCD | – | Charge-Coupled Device |
| CNAME | – | Canonical Name |
| DHCP | – | Dynamic Host Configuration Protocol |
| DNA | – | Deoxyribonucleic Acid |
| DNS | – | Domain Name System |
| DSL | – | Digital Subscriber Line |
| EAP | – | Extensible Authentication Protocol |
| EBGM | – | Elastic Bunch Graph Matching |
| EER | – | Equal Error Rate |
| FAR | – | False Acceptance Rate |
| FRR | – | False Rejection Rate |
| GPG | – | GNU Privacy Guard |
| GPL | – | GNU General Public License |
| HMM | – | Hidden Markov Model |
| HOTP | – | HMAC-based One-time Password |
| HTTPS | – | Hypertext Transfer Protocol Secure |
| CHAP | – | Handshake Authentication Protocol |
| ID | – | Identifier |
| IP | – | Internet Protocol |
| IPv6 | – | Internet Protocol version 4 |
| IPv6 | – | Internet Protocol version 6 |
| L2TP | – | Layer 2 Tunneling Protocol |
| LED | – | Light-Emitting Diode |
| LCD | – | Liquid Crystal Display |
| LDA | – | Linear Discriminant Analysis |
| LDAP | – | Lightweight Directory Access Protocol |
| LTS | – | Long Term Support |
| MAC | – | Media Access Control |
| MD5 | – | Message-Digest algorithm |
| MX | – | Mail Exchanger |

| | | |
|---|---|---|
| NAS | – | Name Server |
| NIR | – | Near-infrared |
| NIST | – | National Institute of Standards and Technology |
| NS | – | Network Access Server |
| NTP | – | Network Time Protocol |
| OATH | – | Initiative for Open Authentication |
| OS | – | Operating System |
| OTP | – | One Time Password |
| PC | – | Personal Computer |
| PAM | – | Pluggable Authentication Modules |
| PAP | – | Password Authentication Protocol |
| PCA | – | Principal Component Analysis |
| PDA | – | Personal Digital Assistant |
| PID | – | Process ID |
| PIN | – | Personal Identification Number |
| PTR | – | Pointer |
| PUK | – | PIN Unlock Key |
| QR | – | Quick Response |
| RADIUS | – | Remote Authentication Dial-In User Service |
| SIM | – | Subscriber Identity Module |
| SixXS | – | Six Access |
| SMS | – | Short Message Service |
| SQL | – | Structured Query Language |
| SSH | – | Secure Shell |
| SSID | – | Service Set Identifier |
| SSO | – | Single Sing-On |
| TAR | – | Tape ARchive |
| TCP | – | Transmission Control Protocol |
| TOTP | – | Time-based One-time Password |
| TTL | – | Time-to-Live |
| UDP | – | User Datagram Protocol |
| USA | – | United States of America |
| USB | – | Universal Serial Bus |
| VLAN | – | Virtual Local Area Network |
| WPA | – | Wi-Fi Protected Access |

# Zoznam obrázkov

# Zoznam tabuliek

# 1 Úvod

Bezpečnosť je v poslednej dobe veľmi diskutovaná téma. Čoraz častejšie sú zaznamenávané prieniky do systémov a zneužitie informácii získané pri týchto prienikoch. Je preto kladený dôraz na čo najväčšiu bezpečnosť systémov obsahujúcich citlivé údaje. Často ale dochádza k problému so znižovaním užívateľského komfortu na úkor bezpečnosti, čo je ale v mnoho prípadoch nevyhnutné. Jednou zo spoľahlivých metód pri zabezpečovaní systémov alebo objektov je autentifikácia na základe biometrie.

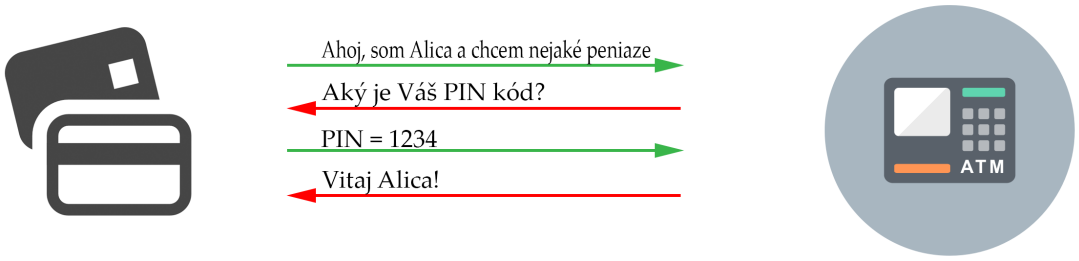
Biometrická autentifikácia sa už využíva nejaký čas, no najčastejšie sme sa s ňou mohli stretnúť pri zabezpečení vstupov do objektu. Postupným zdokonaľovaním výpočtovej techniky si nachádza cestu aj do bežných užívateľských zariadení. Čítačky odtlačkov prstov sú v dnešnej dobe integrované nie len do prenosných počítačov, ale aj mobilných telefónov. Taktiež každé takéto zariadenie disponuje kamerou, ktorá sa môže využívať pri autentifikácii pomocou rozpoznávania tváre.

Obe spomenuté metódy sa tešia veľkej popularite, keďže užívateľovi stačí namiesto zdĺhavého zadávania komplikovaného hesla prejsť prstom po čítačke odtlačkov alebo sa iba pozrieť do kamery. I keď je takýto komfort pre užívateľa na jeho osobnom počítači alebo telefóne dostačujúci, u zariadení ktoré obsahujú citlivé dáta je potrebné zvážiť aj potencionálne bezpečnostné nedostatky týchto biometrických metód.

Preto je vhodné takéto zabezpečenie skombinovať ešte s nejakou ďalšou autentifikačnou metódou. Vznikne tým takzvaná viacfaktorová autentifikácia a práve tej je venovaná táto práca. V teoretickej časti sú podrobne vysvetlené rôzne metódy autentifikácie. Osobitná kapitola je venovaná biometrickým autentifikačným metódam, obzvlášť autentifikácii pomocou odtlačkov prstov, ktorá je využitá v praktickej časti práce. V nej je popísaná presná konfigurácia trojfaktorovej autentifikácie na OS Ubuntu. Navrhnutý systém som napokon overil v laboratórnych podmienkach, čo popisuje posledná časť práce.

# 2 Autentifikácia

Autentifikácia je proces, pri ktorom je overená identita užívateľa, za ktorú sa vydáva. Je to prvý krok pri získavaní prístupu do siete alebo k nejakej službe, po ktorom nasleduje autorizácia a nakoniec účtovanie (Accounting). Existuje viacero možností autentifikácie a budú popísane neskôr v tejto kapitole. Na obrázku 1 je znázornený príklad autentifikácie, ktorý prebieha pri vyberaní peňazí z bankomatu. [1]




Obr. 1: Príklad autentifikácie.

## 2.1 Autentifikačné metódy

Z historického hľadiska sa autentifikačné metódy delili spravidla do troch základných kategórii:

- Čo vieme, napr. heslo, PIN kód a pod.
- Čo máme, napr. platobná karta, čipová karta.
- Kto sme, napr. odtlačok prsta, dúhovka oka a iné.

Avšak tieto kategórie môžu byť mätúce. Heslo nie je až tak niečo čo vieme, ako skôr niečo čo sme si zapamätali. Biometria nie je to kto sme, ale viac to čo nás charakterizuje, ako napríklad naša výška či farba očí. Jednoducho ide o naše fyzické vlastnosti. Namiesto týchto kategórii, sa autentifikačné metódy rozdelili na tri základné metódy, podľa toho, aký druh autentifikácie podporujú. Tieto metódy sú následujúce: [2]

1. ***Heslo*** - využíva k autentifikácii tajomstvo. Užívateľ a autentifikačná autorita zdieľajú medzi sebou toto tajomstvo, ktoré by spravidla nemal poznať nikto iný. Pri tejto metóde je potrebné si zapamätať heslo alebo nejakú skrytú skutočnosť, ako je napríklad meno matky za slobodna. Nevýhodou pri tejto metóde je fakt, že vždy keď je dané tajomstvo odovzdané na autentifikáciu, stáva sa menej bezpečné. Prevažujú tu však výhody, medzi ktoré patria takmer nulové náklady, keďže je heslo najčastejšie zadávané pomocou klávesnice, a taktiež fakt, že heslo nie je možné fyzicky stratiť.

2. ***Token*** - na autentifikáciu využíva nejaký fyzický predmet. Ako token si môžeme predstaviť napríklad klasický kľúč od dverí. Veľkou nevýhodou však je, že keď kľúč stratíme, tak jeho nálezcovi nerobí žiaden problém odomknúť si dvere a vstúpiť do chráneného priestoru. To je dôvod prečo je dnes väčšina digitálnych tokenov chránená ešte ďalšou autentifikačnou metódou. Často býva token spojený s PIN kódom na ochranu proti strate a odcudzeniu. Jednoznačnou výhodou pri tokenoch je, že pri prípadnej strate si je majiteľ vedomí tejto straty a môže teda podniknúť patričné opatrenia proti zneužitiu tokenu.

3. ***Biometria*** - je spôsob autentifikácie, ktorý využíva neschopnosť skopírovať alebo sfalšovať fyzické vlastnosti konkrétnej osoby. Sú to unikátne vlastnosti spojené výhradne iba so svojím majiteľom. Medzi biometrické vlastnosti používané k autentifikácii patrí napríklad odtlačok prsta, dúhovka oka, hlasová biometria alebo podpis. Ďalšou veľkou výhodou pri biometrickej autentifikácii je, že v podstate nemôžeme stratiť to čo využívame na naše overenie (iba v prípade, že by sme prišli napríklad o prst). Na biometrickú autentifikáciu sa využívajú fyzické vlastnosti, ktoré sa ani s vekom nemenia, prípadne len veľmi málo. Táto metóda má však aj nevýhody v podobe vyšších nákladov na dodatočné zariadenia, ktoré sú nutné k biometrickej autentifikácii. Biometria je sama o sebe rozsiahlejšia téma, preto jej bude venovaná celá samostatná kapitola.

## 2.2 Viacfaktorová autentifikácia

Bežný užívateľ využíva mnoho služieb, ktoré sú vo veľkej miere zabezpečené kombináciou prihlasovacieho mena a hesla. Keďže je pre človeka obtiažne si pamätať množstvo hesiel, často si volí rovnaké heslo pre všetky služby. Tým vzniká bezpečnostné riziko, pretože ak útočník získa prihlasovacie údaje k jednej službe, umožní mu to prístup ku všetkým ostatným. Aby sa tomuto zabránilo a bola dosiahnutá vyššia bezpečnosť, je možné skombinovať dve alebo viac autentifikačných metód. Takáto autentifikácia sa nazýva viacfaktorová. Pri tejto autentifikácii je prístup do systému umožnený ak prebehne úspešné overenie všetkých použitých autentifikačných metód.

Najčastejšie je používaná dvojfaktorová autentifikácia, ktorá je často využívaná pri tokenoch, kde poskytuje základnú ochranu proti zneužitiu tokenu. Takýto typ autentifikácie je zobrazený aj na obrázku 1, kde na autentifikáciu je potrebný token (banková karta) a PIN. V poslednej dobe sa dvojfaktorová autentifikácia čoraz častejšie vyskytuje pri zabezpečení napríklad cloudových alebo emailových služieb, v ktorých si užívatelia často ukladajú citlivé informácie. Pri autentifikácii týchto služieb sa najčastejšie používa kombinácia mena a hesla, spolu s jednorázovým heslom.

## 2.3 Autentifikácia heslom

V súčastnosti jednoznačne patrí k najpoužívanejším autentifikačným metódam. Je to spôsobené hlavne tým, že k tejto metóde nepotrebujeme žiadny hardware navyše a celá autentifikácia je

riešená pomocou softwaru. Spravidla sa tu využíva kombinácia užívateľského mena, ktoré reprezentuje identitu užívateľa a hesla, ktorým potvrdzujeme, že identita, za ktorú sa vydávame, skutočne patrí nám. Pokiaľ sa nami zadané údaje zhodujú s údajmi v databáze systému, autentizuje sa naša identita a je nám umožnený vstup do systému.

Aby bolo heslo bezpečné je treba pri jeho výbere dbať na jeho komplexnosť. Ideálne by teda malo obsahovať veľké a malé písmena, číslice a špeciálne znaky. Nevýhodou takto komplexného hesla je predovšetkým jeho ťažká zapamätateľnosť. Taktiež je vhodné používať iné heslo pre každý systém, ktorý využívame. Pre človeka je ale veľmi ťažké si zapamätať množstvo zložitých hesiel. Existujú však techniky, pomocou ktorých sa dá aj ťažké heslo ľahko zapamätať. Napríklad je možné niektoré písmená nahradiť číslicami, čím sa stane heslo oveľa bezpečnejšie, ale stále ľahko zapamätateľné (napr. elektrika = 3l3ktr1k4). Stačí potom zvoliť aspoň jedno písmeno, ktoré budeme písať veľkým písmom a následne podčiarkovníkom identifikujeme systém, pre ktorý budeme heslo používať (napr. 3l3kTr1k4_FB).

Všeobecne sa v organizácii využíva niekoľko nezávislých služieb. Keď chce potom užívateľ niektorú s nich využiť, zobrazí sa mu prihlasovacia obrazovka. Za predpokladu, že užívateľ využíva pre každú službu iné prihlasovacie údaje, musí si pamätať množstvo hesiel. To môže byť pre neho problém a často majú užívatelia tendenciu si tieto hesla niekde napísať. To ale spôsobuje veľké bezpečnostné riziko. Aby sa podobným situáciám zabránilo, bol vyvinutý systém jednotného prihlasovania (SSO). Použitie SSO umožňuje užívateľovi autentifikovať sa do rôznych služieb pomocou jedných prihlasovacích údajov. Ten potom jedným prihlásením získa prístup ku všetkým službám bez toho, aby sa musel znova prihlasovať. [3]

Je niekoľko výhod, ktoré so sebou prináša SSO. Sú to: [3]

- Zvyšuje bezpečnosť systému - prihlasovacie údaje užívateľa sa priamo overujú na centrálnom SSO servery a nie v službe, do ktorej sa snaží užívateľ prihlásiť. Preto ani daná služba nepozná samotné prihlasovacie údaje, ale vie iba o tom, či bol užívateľ úspešne autentifikovaný alebo nie.
- Užívateľský komfort - užívateľ si nemusí pamätať množstvo hesiel a môže využívať rôzne služby bez nutnosti opakovane zadávať prihlasovacie údaje.
- Zjednodušuje správu užívateľov - redukuje množstvo prihlasovacích údajov, ktoré je potrebné spravovať.

Medzi nevýhody patrí fakt, že ak sa útočníkovi podarí získať prihlasovacie údaje do SSO systému, získa prístup ku všetkým službám, ktoré pod SSO spadajú. Preto treba dbať na ochranu prihlasovacích údajov a je vhodné použiť viacfaktorovú autentifikáciu. Taktiež za nevýhodu môžeme považovať závislosť všetkých služieb na SSO. Keď služby stratia dostupnosť na SSO, nebude sa možné do nich prihlásiť.

### 2.3.1 PIN

Do kategórie autentifikácie heslom patrí aj PIN. Autentifikácia založená na PINe môže byť zhrnutá ako komunikácia medzi dvomi entitami, a to medzi užívateľom a systémom, pomocou užívateľského rozhrania. Pri autentifikácii sa overuje, či sa užívateľom zadaný PIN zhoduje s PINom uloženým v databáze autentifikačného systému. Narozdiel od bežného hesla PIN obsahuje iba číslice a typicky pozostáva zo štyroch až ôsmich decimálnych číslic. PIN je často využívaný pri dvojfaktorovej autentifikácii, kedy je v kombinácii s nejakým tokenom. Užívateľ potom pomocou PINu preukazuje, že je oprávnený tento token používať. Tým je daný token chránený proti zneužitiu v prípade odcudzenia alebo straty.

Keďže je PIN pomerne krátky, zväčša jednotnej dĺžky a obsahuje iba číslice, naskytuje sa tu príležitosť na útoky. Pri použití štvormiestneho PINu existuje 10 000 možných kombinácii, čo by prípadnému útočníkovi, pri použití útoku hrubou silou, zabralo relatívne málo času na zistenie správneho PINu. Na ochranu proti tomuto typu útoku slúži limit počtu nesprávne zadaných PIN kódov. Táto ochrana sa implementuje veľmi často. Ako príklad si môžeme zobrať SIM kartu, pri ktorej keď zadáme tri krát nesprávny PIN, dôjde k jej zablokovaniu a na odblokovanie je potrebný PUK kód. Ako ďalší je pozorovací útok, pri ktorom útočník jednoducho odpozoruje, akú kombináciu čísel užívateľ zadal. Proti tomuto útoku neexistuje efektívna ochrana, okrem zakrytia klávesnice na zadanie PINu. [4, 5]

## 2.4 Autentifikácia tokenom

Token je fyzické zariadenie, ktoré môže byť myslené ako prenosné úložisko pre autentifikátor. Môže to byť banková karta, čipová karta, USB kľúč alebo nejaké aktívne zariadenie, ktoré poskytuje heslá meniace sa v závislosti na čase alebo fungujúce na princípe challenge-response (výzva-odpoveď). Keďže sú tokeny určené pre bezpečnostnú aplikáciu, je väčšina z nich zabezpečená proti kopírovaniu a falšovaniu. Tokeny sú vo väčšine prípadov zabezpečené dvojfaktorovou autentifikáciou, a teda aby sme získali plnohodnotný prístup k tokenu, je potrebné ešte poznať heslo alebo sa overiť pomocou biometrie. Ako heslo sa najčastejšie používa PIN. Token môže v sebe obsahovať heslo, ktoré si užívateľ zvolí alebo pre plné využitie potenciálu tokenu, môže byť do neho uložené aj oveľa dlhšie kódové slovo, ktoré je pre človeka nemožné si zapamätať. Za kódové slovo môžeme považovať klasické heslo, ktoré však bolo strojovo vygenerované a uložené do tokenu. Tým pádom môže byť dlhšie a viac náhodnejšie ako človekom vytvorené heslo, čo výrazne zvyšuje aj jeho bezpečnosť. Existujú dva typy tokenov:

- **Pasívne** - napr. čipová karta, USB kľúč
- **Aktívne** - napr. generátor jednorázového hesla

Pri pasívnych tokenoch je kódové slovo zvyčajne uložené na magnetickom prúžku, čipovej karte alebo prípadne USB kľúči. Pri tomto type je kódové slovo statické. Aktívne tokeny naj-

častejšie obsahujú mikroprocesor, ktorý pomocou algoritmov počíta jednorázové heslo. V niektorých prípadoch slúži aktívny token aj na vykonávanie kryptografických výpočtov, ktoré realizujú šifrovanie a dešifrovanie. Čipová karta môže byť využitá pri challenge-response autentifikácii s využitím kryptografického procesora na karte. Takáto autentifikácia poskytuje vysoké zabezpečenie, ale vyžaduje pokročilejšiu čítačku kariet a taktiež nie každá čipová karta má v sebe zabudovaný mikroprocesor. Ak ho karta má, radí sa k aktívnym tokenom, ale ak slúži iba na uloženie hesla alebo kódového mena, radí sa k pasívnym tokenom.

Hlavnou výhodou tokenov je ich bezpečnosť. Keďže môžu generovať alebo mať v sebe uložené veľmi dlhé kódové slovo, je ich prelomenie omnoho zložitejšie. Token, ktorý poskytuje kódové slovo zložené z dvanástich číslic, má $10^{12}$ možných kombinácii. Množstvo týchto kombinácii sa nazýva priestor kľúča (z angl. *keyspace*) a je žiadúce aby bol čo najväčší. Čím väčší priestor kľúča, tým je token viac chránený proti útoku hrubou silou. Narozdiel od autentifikácie heslom, kedy užívateľ nemusí ani tušiť o prelomení hesla, si je užívateľ pri strate alebo odcudzení tokenu vedomí tejto straty a môže tak vykonať kroky proti zneužitiu tokenu.

Tokeny so sebou však prinášajú aj niekoľko nevýhod. Ako hlavnú nevýhodu môžeme považovať nutnosť vlastniť dodatočný hardware. Každý token potrebuje čítačku alebo port aby mohol sprostredkovať informácie z tokenu. Ide napríklad o čítačku čipových kariet a USB port. Keďže sa tokeny zvyčajne používajú s dvojfaktorovou autentifikáciou, a to najčastejšie v kombinácii s PINom, vyskytujú sa tu podobné problémy s memorovaním ako u hesiel. Je ale potrebné poznamenať, že štvormiestny PIN sa dá zapamätať omnoho ľahšie ako komplexné heslo. Ako poslednú nevýhodu by som spomenul, že užívateľ musí myslieť na to aby mal token fyzicky pri sebe. [2, 6]

### 2.4.1 Jednorázové heslo

Medzi tokeny sa radí aj takzvané jednorázové heslo. Narozdiel od bežných hesiel, ktoré sú pri každom prihlásení rovnaké, jednorázové heslo sa mení pri každom použití. Využíva sa prevažne pri dvojfaktorovej autentifikácii a potvrdzuje identitu užívateľa, za ktorú sa vydáva. Jednorázové heslo poskytuje ochranu proti odpočúvaniu média. Keďže je pre každú reláciu iné heslo, útočníkom zachytené heslo by už neviedlo k úspešnej autentifikácii. Obtiažna pri tejto metóde je generácia a distribúcia jednorázového hesla. Existujú dva základné princípy, na ktorých funguje systém jednorázového hesla: [6]

- Zoznam, ktorý je náhodne vygenerovaný, pričom jedna kópia zostáva u užívateľa a jedna je uložená v systéme.
- Challenge-response systém, pri ktorom je heslo vygenerované na základe požiadavky od užívateľa a overené systémom.

Z týchto princípov vychádzajú jednotlivé metódy použitia, ktoré budú ďalej popísané.

#### 2.4.1.1 Grid karta

Za najjednoduchšiu metódu sa dá považovať grid karta. Táto karta obsahuje náhodne vygenerované čísla, znaky alebo ich kombináciu, ktoré sú usporiadané do riadkov a stĺpcov. Pri použití v dvojfaktorovej autentifikácii sa najčastejšie vyskytuje v kombinácii s heslom. Užívateľ najprv zadá svoje prihlasovacie meno a heslo, a potom, za predpokladu že ich zadal správne, je vyzvaný zadať náhodne vybranú bunku z grid karty. Ak sa zadaný údaj zhoduje z údajom v systéme, je užívateľ úspešne autentifikovaný. Príklad grid karty je zobrazený na obrázku 2.

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| **1** | AZZR | CMNZ | WZMZ | EXEV | RTRW | DYQV | QDHE |
| **2** | TXGU | NWDH | NKTY | GYZO | HLXO | YQZF | VHRQ |
| **3** | CQWX | UEVY | GDIO | OOJE | ICXS | JCHS | AQLV |
| **4** | CJLF | TAJD | BGJR | XUZN | SAEK | KZXH | TUZD |
| **5** | JEQL | CHYO | QQYV | SACX | IYCO | GHIU | GILF |
| **6** | IUGJ | WSDW | UXBA | RVWP | EEGB | USZG | NFGB |
| **7** | ZRCL | PQWZ | QZDF | GNKT | EHIE | QDIS | XHIU |

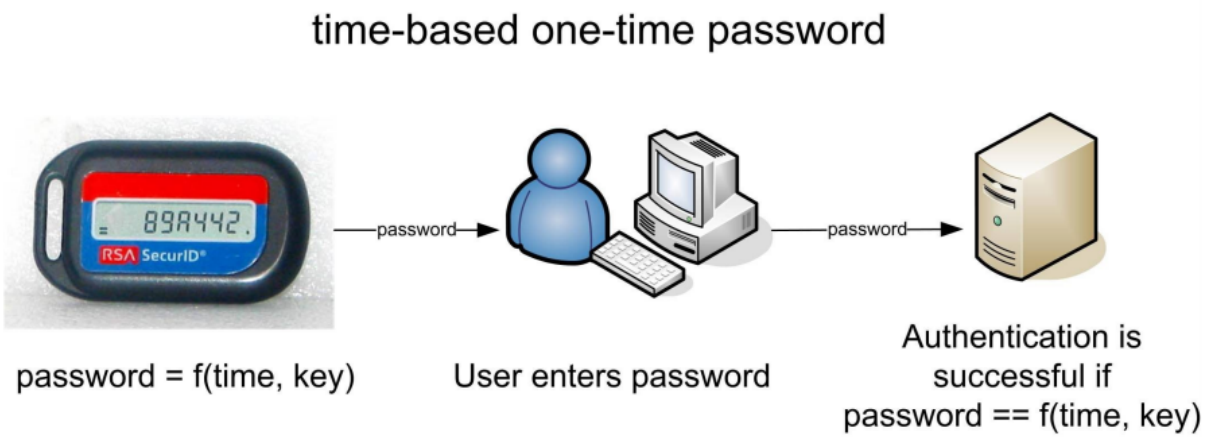
Obr. 2: Príklad grid karty.

Veľkosť karty určuje počet buniek, ktoré obsahuje. Zvyčajne má rozmery bežnej platobnej karty, a je teda ľahko uskladniteľná. Taktiež je táto metóda pomerne lacná a poradí si s ňou aj bežný užívateľ. Navyše pri strate karty je jej výmena relatívne jednoduchá. Táto metóda bola v minulosti často využívaná pri prihlasovaní do internetového bankovníctva. [6]

#### 2.4.1.2 Časovo závislé

V týchto systémoch sa jednorázové heslo mení v závislosti na čase, a to pomocou algoritmu, ktorý je známy systému a autentifikačnému zariadeniu, ktoré vlastní užívateľ. Jednorázové heslo sa generuje pravidelne vo vopred definovanom časovom intervale. Algoritmus vygeneruje heslo pomocou funkcie z kľúča a aktuálneho času. Kľúč býva prednastavený pri výrobe zariadenia, prípadne je nastavený pri inicializácii softvérového tokenu. Aby sa heslá vytvorené tokenom zhodovali s tými, ktoré vytvorí systém pri autentifikácii, je potrebné zaistiť časovú synchronizáciu. To môže byť niekedy obtiažne, a preto autentifikačný server zvyčajne vypočíta niekoľko hesiel s malými časovými odchýlkami. To spôsobí, že systém akceptuje niekoľko hesiel naraz. Princíp časovo závislého jednorázového hesla je zobrazený na obrázku 3.

Zvyčajne sa jedná o malé zariadenie s LCD displejom, ktorý zobrazuje aktuálne jednorázové heslo. Taktiež existujú softwarové tokeny, ktoré si užívateľ nainštaluje na nejaké prenosné zariadenie, napríklad smartfón. Pri softvérovom riešení je potrebné najprv inicializovať token. Toto je riešené najčastejšie pomocou QR kódu, ktorý vygeneruje systém a následne je spraco-

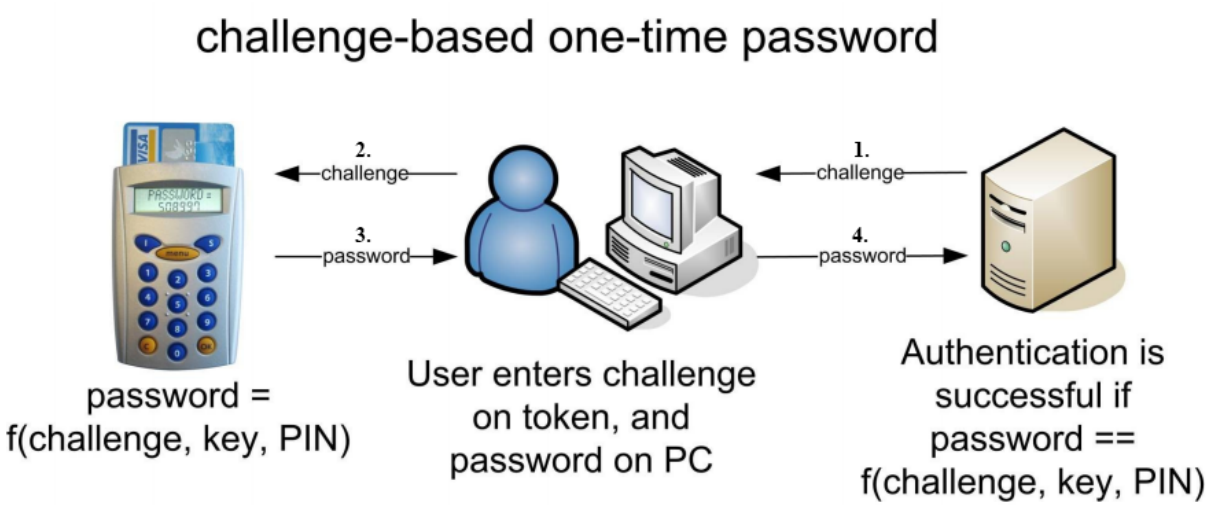



Obr. 3: Princíp časovo závislého jednorázového hesla. [6]

vaný pomocou softwaru v smartfóne. Použitím tohoto spôsobu sa vyhneme posielaniu dôverných správ cez nezabezpečenú sieť. [6, 7]

#### 2.4.1.3 Challenge-Response

Keď sa užívateľ snaží pri tejto metóde autentifikovať, systém vygeneruje náhodnú výzvu. Užívateľ odomkne svoj token pomocou PIN kódu a zadá výzvu. Token následne vypočíta odpoveď a zobrazí ju na displeji. Užívateľ pošle výsledok naspäť systému ako jeho odpoveď na výzvu a ak sa zhoduje s heslom vygenerovaným systémom, je užívateľ úspešne autentifikovaný. Tým dosiahneme toho, aby sa cez sieť posielala iba náhodná výzva a šifrovaný výsledok. Neposiela sa teda ani užívateľov PIN alebo kľúč, čo zabezpečuje autentifikáciu proti odpočúvaniu na médiu. Tento princíp je zobrazený na obrázku 4.

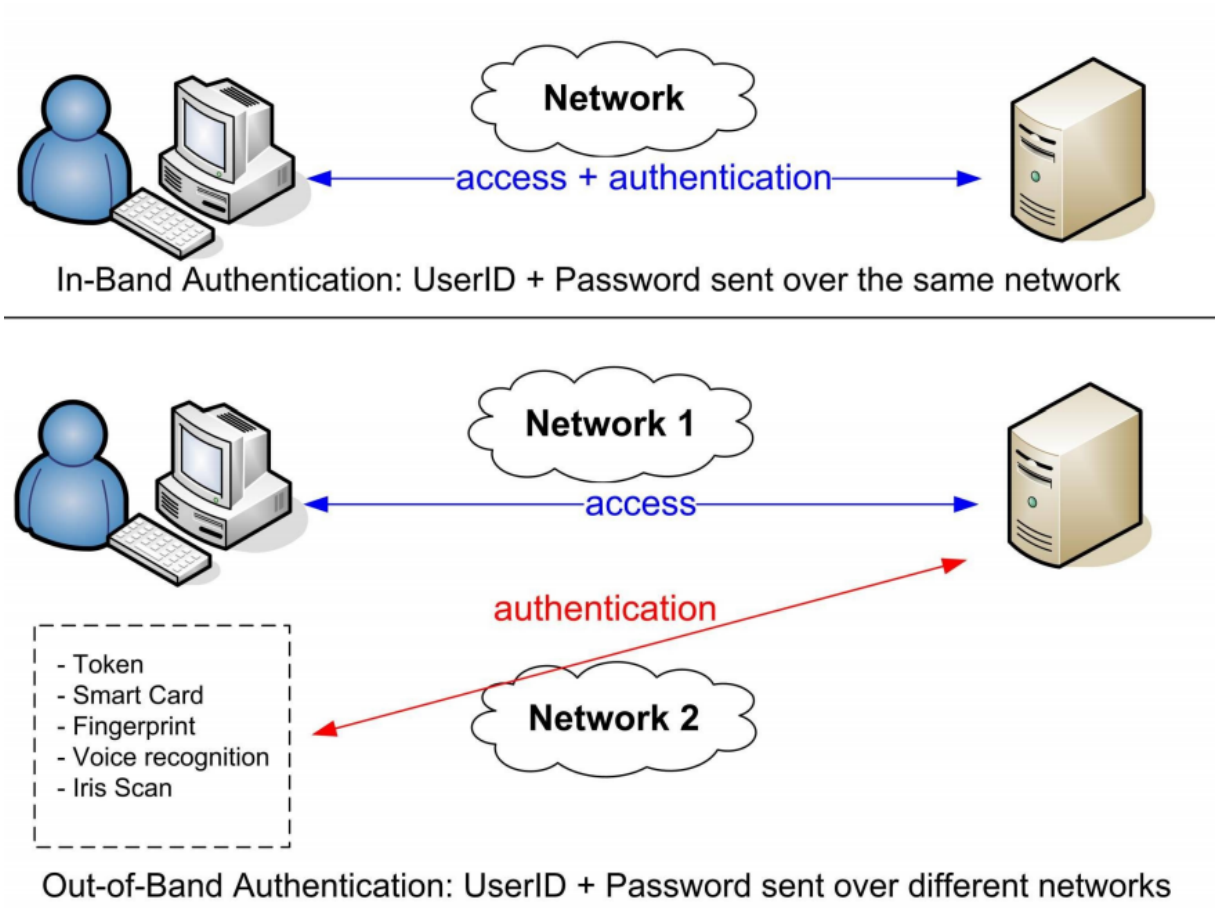



Obr. 4: Princím systému založenom na Challenge-Response. [6]

Narozdiel od časovo závislého systému, je tu čas nahradený výzvou. Challenge-Response systém využíva najčastejšie ako token čipovú kartu, ktorá je chránená PINom. Funkcia na výpočet hesla používa ako vstup výzvu a kľúč. Tieto systémy sa považujú za o čosi málo bezpečnejšie ako časovo závislé systémy, keďže výzva je riadená na strane autentifikačného servera a token je dodatočne chránený PINom. [6]

#### 2.4.1.4 Out-of-band

Pri takzvanom out-of-band prenose je použitý iný prenosový kanál ako ten, ktorý užívateľ využíva na prenos ostatných dát. Tento samostatný kanál poskytuje ďalšiu vrstvu ochrany, keďže by potencionálny útočník musel zachytávať obidva prenosné kanály. Ako out-of-band prenos môže byť využitý napríklad email, volanie alebo SMS správa na mobilný telefón. Často je táto metóda využívaná na potvrdzovanie transakcií v internetovom bankovníctve. Keď banka obdrží požiadavku na transakciu, pošle klientovi SMS správu, ktorá obsahuje podrobnosti transakcie a jednorázové heslo. Týmto heslom sa potom užívateľ autentifikuje a potvrdí transakciu. Znázornenie tejto metódy je na obrázku 5. [6]




Obr. 5: Princíp systému založenom na out-of-band prenose. [6]

# 3 Biometrická autentifikácia

Pri biometrickej autentifikácii sa využívajú unikátne vlastnosti konkrétnej osoby. Existuje mnoho autentifikačných metód s využitím biometrie človeka a z nich budem popisovať tie najčastejšie využívané pri autentifikácii. Najprv je však potrebné popísať základný princíp fungovania tejto metódy autentifikácie.

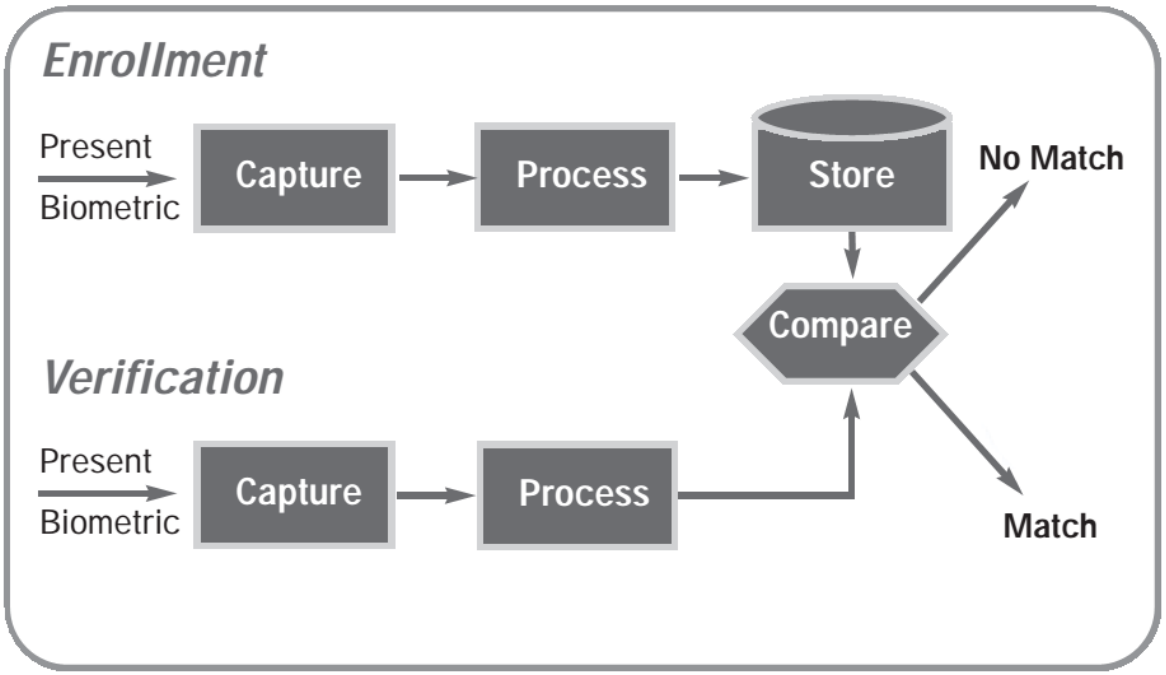
Pri biometrickej autentifikácii sa porovnáva zaznamenaná a uložená biometrická vzorka, voči novo zachytenej vzorke (napríklad tej, ktorá je použitá na autentifikáciu). Na to aby sa mohli tieto vzorky porovnať, je najprv potrebné vykonať tieto tri kroky: [29]

1. **Zaznamenanie** - biometrická vzorka je zaznamenaná pomocou snímacieho zariadenia, ako napríklad čítačka odtlačkov prsta.

2. **Spracovanie** - z biometrickej vzorky sa vyberú rozlišovacie charakteristické rysy, ktoré sú skonvertované do takzvaného biometrického etalónu.

3. **Uloženie** - spracovaný etalón je uložený na ukladacie médium pre neskoršie porovnanie pri autentifikácii. Ako ukladacie médium je možné použiť samotné snímacie zariadenie, databázu, prípadne prenosné zariadenie, ako napríklad čipová karta. Pôvodnú biometrickú vzorku nie je možné získať z uloženého etalónu.

Proces zaznamenávania a následného porovnania pri autentifikácii zobrazuje obrázok 6.




Obr. 6: Proces zaznamenávania a porovnania biometrického etalónu. [29]

Pri navrhovaní biometrického autentifikačného systému je treba poznať jeho výkonnosť. Tá sa určuje na základe viacero štatistických koeficientov. Medzi tie najpodstatnejšie pri určovaní efektívnosti môžeme zaradiť tieto tri: [30, 31]

1. **False Rejection Rate (FRR)** - určuje pravdepodobnosť, že biometrický systém zlyhá pri autentifikácii registrovanej osob, čiže odmietne oprávneného užívateľa. Z bezpečnostného hľadiska je tento koeficient zanedbateľný, no znižuje užívateľský komfort, keďže užívateľ musí opakovať autentifikačný proces. Tento koeficient sa určuje zo vzťahu:

   $$FRR = \frac{N_{FR}}{N_{EIA}} \cdot 100\,[\%]$$

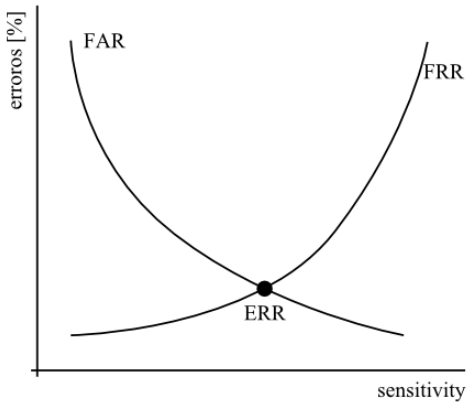
   $N_{FR}$ - počet chybných odmietnutí
   $N_{EIA}$ - počet všetkých pokusov oprávnených osôb o identifikáciu

2. **False Acceptance Rate (FAR)** - reprezentuje pravdepodobnosť chybnej úspešnej autentifikácie. To znamená, že systém úspešne autentifikuje neoprávnenú osobu. Tento koeficient je kľúčový pri určovaní bezpečnosti systému a určuje sa zo vzťahu:

   $$FAR = \frac{N_{FA}}{N_{IIA}} \cdot 100\,[\%]$$

   $N_{FA}$ - počet chybných prijatí
   $N_{IIA}$ - počet všetkých pokusov neoprávnených osôb o identifikáciu

3. **Equal Error Rate (EER)** - vyjadruje hodnotu, pri ktorej sú koeficienty FAR a FRR rovné. Ideálny systém je, keď hodnota FRR = FAR = 0, to však nie je možné v reálnych podmienkach dosiahnuť. Pre bezpečný systém by mala byť hodnota FAR čo najnižšia, to ale v mnohých prípadoch znižuje užívateľský komfort. Preto je pri ERR systém v ideálnej rovnováhe medzi bezpečnosťou a užívateľským komfortom. Grafické zobrazenie ERR je na obrázku 7.




Obr. 7: Zobrazenie ERR v závislosti na FAR a FRR. [32]

Ako už bolo spomenuté biometrických metód pre identifikáciu je mnoho. Všetky sa však dajú zaradiť do jednej z dvoch skupín, a to fyziologické a behaviorálne. V následujúcej kapitole bude popísaná metóda autentifikácie pomocou odtlačku prsta, pretože práve tu využívam v praktickej časti práce. Ostatné, ktoré sa najčastejšie používajú pri autentifikácii, sú v prípade záujmu čitateľa popísané v prílohe A. [33]

## 3.1 Fyziologické metódy

Pri fyziologickej biometrii sa využíva špecifický tvar alebo štruktúra časti ľudského tela, ktorá je pre každého človeka jedinečná. Do tejto kategórie patrí identifikácia na základe odtlačku prsta, geometrie ruky, štruktúry žíl na ruke, geometrie tváre, sietnice, dúhovky, ale napríklad aj na základe DNA.

### 3.1.1 Odtlačok prsta

Identifikácia na základe odtlačku prsta je bezpochyby najčastejšie sa vyskytujúca biometrická metóda. Je to vďaka unikátnosti odtlačku a jeho konzistentnosťou, ktorá sa v čase nemení. Odtlačky prsta sa na identifikáciu využívajú už viac ako storočie, pričom v nedávnej dobe došlo k automatizácii celého procesu, vďaka pokrokom vo výpočtovej technike. [34]

Vnútorná strana našich rúk je pokrytá takzvanými „výbežkami“ a „ryhami“. Tieto papilárne línie (výbežky) sa využívajú pri biometrickej identifikácii. Vzor, ktorý tvoria papilárne línie je považovaný za unikátny a nezameniteľný. Dokonca aj u identických dvojčiat je možné použiť odtlačok prsta na ich rozoznanie. Pri porezaní prsta alebo inom zranení dôjde k dočasnej zmene vzoru papilárnych línii, ktorá sa obnoví po tom, ako sa zranenie zahojí. Ak však bolo poranenie na najvnútornejšej vrstve pokožky, zanechá na prste jazvu, ktorá natrvalo zmení vzor papilárnych línii. [33, 35]

Prvým krokom pri autentifikácii pomocou odtlačku prsta je zosnímanie a digitalizácia odtlačku pre ďalšie spracovanie. Hlavným dôvodom populárnosti odtlačkov prstov je dostupnosť relatívne lacných senzorov, ktoré sú schopné pomerne rýchlo zosnímať odtlačok prsta s minimálnou alebo žiadnou potrebou na zásah od užívateľa. Vďaka kompaktným rozmerom takýchto senzorov, bývajú často zabudované do užívateľských zariadení ako sú napríklad notebooky alebo mobilné telefóny. Väčšina senzorov na získanie odtlačku prsta je založená buď na optických alebo kapacitných technológiách. Medzi najčastejšie používané môžme zaradiť tieto z nich: [35]

- **Optické** - pri tejto technológii je využívaná sklenená doska, zdroj svetla (LED alebo laser) a CCD kamera, pomocou ktorej získame výsledný obrázok odtlačku prsta. Pri snímaní sa prst položí na jednu stranu sklenenej dosky. Zdroj svetla a CCD kamera sú umiestnené na druhej strane dosky. Zdroj svetla osvetlí pod určitým uhlom sklo, od ktorého sa odrazí a je zachytené kamerou. Svetlo dopadajúce na výbežky prsta je náhodne rozptýlené, a tým sa zobrazia na obrázku ako tmavé miesta, zatiaľ čo ryhy spôsobia celkový vnútorný odraz

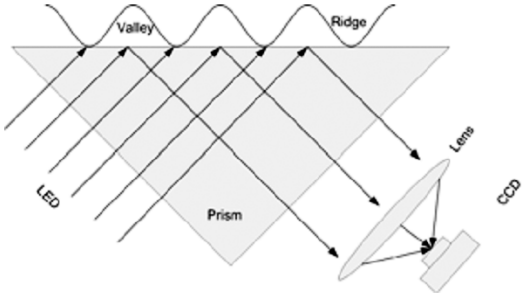
a zobrazia sa ako svetlé miesta. Pri tejto technológii je ťažké dosiahnuť kompaktných rozmerov, keďže ohnisková vzdialenosť malých šošoviek môže byť veľmi veľká. Taktiež môže dochádzať k znečisteniu sklenenej dosky, a tým k neúspešnej autentifikácii. Princíp optického snímania odtlačkov je zobrazený na obrázku 8.

- **Kapacitné** - tento typ senzorov je najčastejšie využívaný vďaka kompaktným rozmerom, ktoré umožňujú tento senzor zabudovať do rôznych prenosných zariadení ako je mobilný telefón alebo periférie počítača. Kapacitný senzor v základe pozostáva z poľa elektród, ktoré sú zložené s tisícov kapacitných dosiek zabudovaných v čipe senzora. Pokožka prsta sa chová ako druhá elektróda, čím vytvára miniatúrny kondenzátor. Medzi povrchom prsta a doskami na čipe vzniká malý elektrický náboj. Veľkosť elektrického náboju závisí na vzdialenosti medzi povrchom prsta a kapacitných dosiek. Preto na výbežkoch a ryhách vzniká rôzna kapacita, pričom na výbežkoch je väčšia ako na ryhách. Rozdiel v kapacitách je základom fungovania tohoto senzoru. Táto technika je veľmi náchylná na elektrostatický výboj, ktorý môže poškodiť senzor. Preto je potrebné zabezpečiť správne uzemnenie aby sme sa vyhli tomuto problému.

- **Ultrazvukový odraz** - táto technológia je založená na posielaní akustických signálov na končeky prsta a zachytávanie odrazeného signálu. Odrazený signál je použitý na výpočet obrázka odtlačku prsta a následne vzoru samotných papilárnych línii. Senzor pozostáva z dvoch základných komponentov. Prvým je vysielač, ktorý generuje krátke akustické pulzy, a druhý je prijímač, ktorý detekuje odpovede získané pri odraze pulzov z povrchu prsta. Táto metóda vytvára podpovrchový obrázok plochy prsta, vďaka čomu je odolný na znečistenie, ktoré môže viesť k vizuálnemu znehodnoteniu odtlačku prsta. Tento typ senzorov je však veľmi drahý a preto sa príliš často nevyužíva.

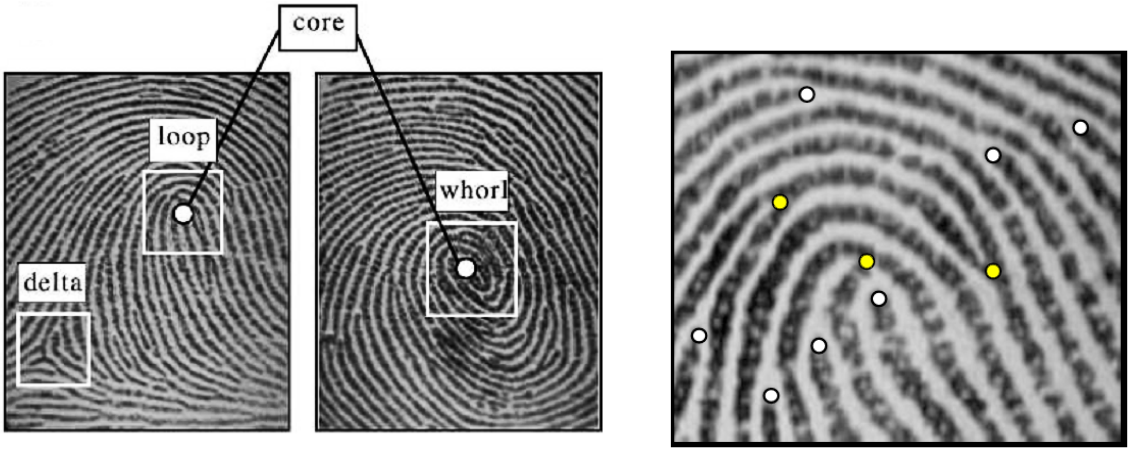
- **Piezoelektrický efekt** - jedná sa o senzory citlivé na tlak, ktoré produkujú elektrický signál keď na ne pôsobí mechanické namáhanie. Povrch senzora je vyrobený z nevodivého dielektrického materiálu, ktorý pri tlaku, vytvorenom pri priložení prsta, generuje malé množstvo elektrického napätia. Veľkosť vygenerovaného napätia závisí na veľkosti tlaku, pod ktorým sa prst dotýka povrchu senzora. Keďže výbežky a ryhy sa nachádzajú v rozličných výškach, budú na povrch pôsobiť pod rôznym tlakom, a teda generovať aj napätie rôznej veľkosti. Problémom pri týchto senzoroch býva ich nízka citlivosť, pri ktorej nie sú schopné zaznamenať presný vzor papilárnych línii.

- **Teplotný rozdiel** - senzor, ktorý využíva túto technológiu je vyrobený z pyro-elektrického materiálu, ktorý generuje napätie na základe teplotných rozdielov. Tento rozdiel vzniká keď sú dva povrchy spojené a vznikne medzi nimi kontakt. Výbežky na prste sú v priamom kontakte s povrchom senzora, zatiaľ čo ryhy sú od tohoto povrchu vzdialené. Tým vzniká teplotný rozdiel medzi výbežkami a ryhami, vďaka čomu je možné vytvoriť obrázok odtlačku prsta.




Obr. 8: Princíp optického snímania odtlačkov prsta. [35]

Kvalita skenera odtlačkov prsta, veľkosť jeho snímacej plochy a výsledné rozlíšenie získaného odtlačku majú veľký vplyv na výkon rozpoznávacieho algoritmu pre odtlačky. Vo výslednom obrázku sú papilárne línie (výbežky) tmavé, zatiaľ čo ryhy na prste svetlé. Tieto dva prvky vytvárajú štyri základné charakteristické tvary, ktoré sú skúmané pri rozpoznávaní odtlačku prsta a nazývajú sa oblúk, slučka (loop), vír (whorl) a delta. Tieto tvary bývajú označované ako singularity. Nachádza sa tu aj miesto nazývané jadro (core), ktoré býva často využívané na zarovnanie odtlačku prsta pri jeho skenovaní. Všetky spomenuté tvary a jadro zobrazuje obrázok 9a. [36]



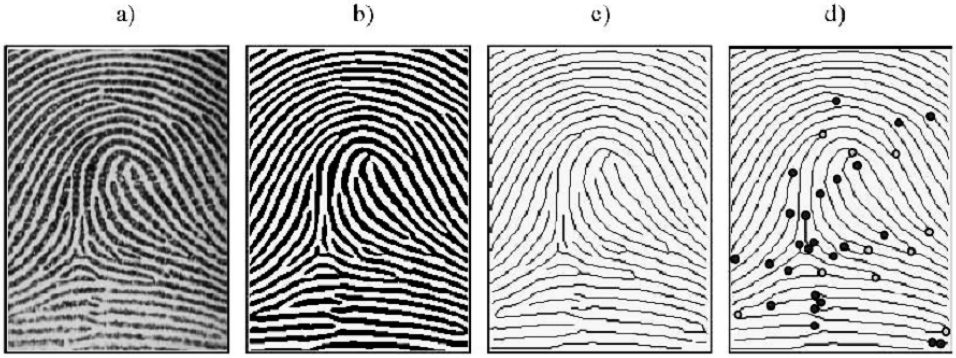

(a) Jadro a singularity na prste.

(b) Markanty zobrazené na vzorovom odtlačku prsta.

Obr. 9: Skúmané body na odtlačku prsta. [36]

Ďalej sa v odtlačku skúmajú takzvané markanty, ktoré určujú akým spôsobom sú papilárne línie ukončené. Hoci existuje viacero typov markant, zvyčajne sa pri automatickom rozpoznávaní využívajú iba dva základné typy, ktoré sa dajú ľahko rozlíšiť s vysokou presnosťou. Ide o náhle ukončenie papilárnej línie alebo jej rozdelenie do dvoch línii. Obidva typy zobrazuje obrázok 9b, na ktorom sú zakončenia zobrazené bielou a rozdvojenia žltou farbou.

Na získanie markant z nasnímaného odtlačku prsta v podobe čierno-bieleho obrázka (a), je potrebné tento obrázok skonvertovať do binárnej podoby (b) pomocou procesu nazývaného binarizácia. Binárny obrázok prejde potom takzvaným stenčovacím procesom, pri ktorom je hrúbka papilárnych línii zmenšená na jeden pixel (c). Nakoniec sa s pomocou algoritmu nájdu zakončenia a rozdvojenia papilárnych línii (d). Tieto štyri kroky sú v postupnom slede zobrazené na obrázku 10. [36]




Obr. 10: Postupný proces získavania markant. [36]

Pri rozpoznávaní odtlačkov sa najčastejšie využívajú dve metódy. Prvou z nich je metóda založená na rozpoznávaní markant. Je to najpopulárnejšia a najrozšírenejšia metóda rozpoznávania odtlačkov. Markanty sú uložené v podobe množiny bodov v dvojrozmernej rovine. Každý bod pozostáva z dvoch súradníc na určenie jeho polohy a z uhla, ktorý určuje orientáciu markanty. Druhou je metóda založená na základných tvaroch papilárnych línii, ktoré sú nazývané singularity. Táto metóda nevyžaduje takú vysokú kvalitu oskenovaného odtlačka ako metóda rozpoznávania markant. Väčšinou sa však táto metóda využíva v kombinácii s detekciou markant na zvýšenie presnosti daného systému. [36]

Čo sa týka výkonnosti tohoto biometrického systému, hodnota koeficientu FRR sa pohybuje na hodnote <$1.0\%$ a hodnota FAR v rozmedzí od $0.00001\%$ do $0.0001\%$ v závislosti na použitom type skenera. Pri takýchto nízkych hodnotách dôležitých koeficientoch môžeme povedať, že miera spoľahlivosti takéhoto systému je vysoká. Čas verifikácie sa pohybuje v rozmedzí $0.2 - 1$ sekunda.[30]

Nevýhodou pri tejto autentifikačnej metóde je, že sa dá pomerne ľahko oklamať priložením napodobeniny prsta. Taktiež prípadne zranenie na prste môže viesť k neúspešnej autentifikácii oprávneného užívateľa. [36]

## 3.2 Behaviorálne metódy

Behaviorálna biometria sa zameriava na to ako niečo vykonávame. Využíva vlastnosti, ktoré nie je možné napodobniť inou osobou. Jedná sa napríklad o identifikáciu na základe dynamiky podpisu, chôdze, stláčania kláves alebo aj charakteristiky hlasu.

# 4 RADIUS protokol a PAM moduly

## 4.1 RADIUS protokol

Pod skratkou RADIUS sa skrýva pomerne dlhý názov Remote Authentication Dial In User Service. Je to sieťový protokol, ktorý slúži na vzdialené zabezpečenie autentifikácie, autorizácie a účtovania (AAA). Autentifikácia slúži na overenie užívateľa, či je naozaj ten za koho sa vydáva, autorizácia umožňuje tomuto užívateľovi používať špecifické služby a účtovanie zaznamenáva používanie týchto služieb. RADIUS protokol je popísaný v štandarde RFC2865, pričom účtovanie je popísané v samostatnom štandarde RFC2866. Vďaka týmto štandardom je RADIUS protokol voľne dostupný pre všetkých vývojárov a výrobcov, čo zapríčinilo, že sa RADIUS stal rozšíreným protokolom na poskytovanie AAA v TCP/IP sieťach. S RADIUS protokolom sa je možné najčastejšie stretnúť pri overovaní užívateľa na VPN koncentrátore, pripojovaní na bezdrôtovú sieť, ktorá využíva zabezpečenie WPA2 Enterprise alebo pri pripojovaní užívateľa na internet pomocou DSL. [1]

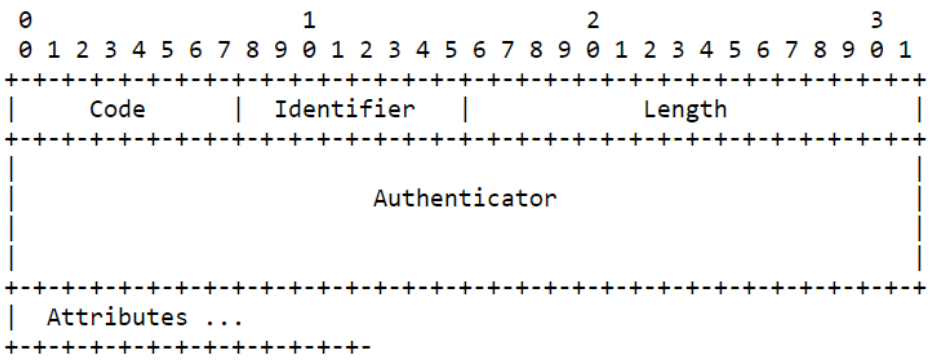
RADIUS je protokol založený na modelu klient-server, kde NAS vystupuje ako klient. NAS poskytuje prístup užívateľa do siete a môže to byť napríklad bezdrôtový prístupový bod, DSLAM alebo VPN koncentrátor. Užívateľské zariadenie vždy komunikuje iba s NAS a ten posúva požiadavky ďalej na RADIUS server, ktorý tieto požiadavky od užívateľa spracuje a posiela príslušné odpovede naspäť na NAS. Správy prenášané medzi serverom a klientom sú autentifikované na základe zdieľaného hesla. Navyše akékoľvek užívateľské heslá sú šifrované, čo slúži ako ochrana proti odpočúvaniu na nezabezpečenom médiu. RADIUS podporuje rôzne druhy autentifikačných mechanizmov ako sú napríklad PAP, CHAP, EAP alebo unixové prihlasovanie. Existuje niekoľko programov, ktoré vuyžívajú RADIUS na svoju činnosť. Medzi najznámejšie patria FreeRADIUS, Radiator, IAS od Microsoftu a ACS od Cisca. [40, 41]

RADIUS protokol umožňuje RADIUS serveru vystupovať ako klient pred iným RADIUS serverom, čo z neho robí takzvaný proxy server. Viacero takýchto proxy serverov za sebou tvorí reťazec. V takomto systéme sú využité doménové mená jednotlivých užívateľov. Užívateľské meno je od názvu domény oddelené rozdeľovacím znakom. Najčastejšie to býva „@“ (`bob@firma.cz`) alebo pri Windows systémoch „\“ (FIRMA\bob). Na základe tohto doménového mena sa RADIUS server rozhoduje, či danú požiadavku spracuje alebo ju prepošle na iný RADIUS server, ktorý obsluhuje požiadavky danej domény. V takom prípade musí mať cieľový RADIUS server definovaný zdrojový RADIUS server ako NAS klienta. [1]

### 4.1.1 Formát RADIUS Paketu

Pakety, ktoré si medzi sebou vymieňajú klient a server sú zapuzdrené pomocou UDP protokolu. RADIUS využíva pri činnosti dva porty tohoto protokolu. Prvým je 1812, na ktorom spracováva pakety posielané pri autentifikácii a druhým je 1813, ktorý slúži na spracovanie

paketov pre účtovanie. Formát dátového paketu, tak ako je definovaný v štandarde RFC2865, je zobrazený na obrázku 11. [1, 40]




Obr. 11: Formát dátového paketu protokolu RADIUS. [40]

Každý paket je identifikovaný pomocou kódu v poli **Code**, ktorý má veľkosť 8 bitov a jeho hodnota určuje o aký typ RADIUS paketu sa jedná. Všetky typy paketov a ich kódy sú v tabuľke 1. Pri autentifikácii sa vyskytujú tieto štyri typy paketov: [1, 41]

- **Access-Request** -obsahuje informácie o užívateľovi, v ktorých je obsiahnuté jeho prihlasovacie meno, údaje potrebné na autentifikáciu a služba, ku ktorej žiada prístup. Taktiež sú tu zahrnuté informácie o NAS klientovi, ako je jeho doménové meno, MAC adresa alebo SSID v prípade, že sa jedná o prístupový bod. Táto správa je vždy posielaná autentifikačným protokolom PAP, CHAP alebo EAP.
- **Access-Accept** - v prípade tejto odpovede, je užívateľovi poskytnutý prístup k požadovanej službe, ktorú vyžaduje od NAS. Ak potrebuje užívateľ využiť nejakú inú službu, na RADIUS serveri sa overí, či má oprávnenie ju používať. Medzi najčastejšie služby, ktoré NAS poskytuje užívateľovi patrí pridelenie statickej alebo dynamickej IP adresy, TTL pre danú relácie, nastavenie akéhokoľvek L2TP tunela a pridelenie užívateľa do VLAN.
- **Access-Reject** - túto odpoveď posiela RADIUS server keď zamietne užívateľovi prístup k požadovanej službe.
- **Access-Challenge** - takáto odpoveď od servera vyžaduje od užívateľa, aby poskytol ďalšie informácie potrebné k autentifikácii.

Ďalšie pole v dátovom pakete RADIUSu, ktoré má tiež 8 bitov, je **Identifier**. Ako už názov napovedá, jedná sa o identifikátor, na základe ktorého klient rozpozná, ktoré odpovede od servera má priradiť ku ktorej požiadavke. Keďže RADIUS protokol posiela pakety pomocou UDP protokolu, má v sebe implementovaný mechanizmus na opakované zaslanie požiadavky. Keď klient opakovane zašle požiadavku na server, identifikátor ostane rovnaký.

| Kód (Decimálne) | Typ paketu | Posiela ho |
|---|---|---|
| 1 | Access-Request | NAS |
| 2 | Access-Accept | RADIUS server |
| 3 | Access-Reject | RADIUS server |
| 4 | Accounting-Request | RADIUS server |
| 5 | Accounting-Response | NAS |
| 11 | Access-Challenge | RADIUS server |
| 12 | Status-Server (Experimental) | |
| 13 | Status-Client (Experimental) | |
| 255 | Reserved | |

Tabuľka 1: Kódy používané na identifikáciu RADIUS paketov. [1]

Za identifikátorom nasleduje pole s označením **Length**, ktoré ma veľkosť 16 bitov a určuje celkovú veľkosť paketu. Maximálna povolená veľkosť RADIUS paketu je 4096 bajtov.

Predposledné pole má názov **Authenticator** a má veľkosť 128 bitov. Jeho obsah sa líši v závislosti na tom, či ide o požiadavku alebo odpoveď. Taktiež závisí od typy paketu, v ktorom je obsiahnutý. Ak ide o požiadavku, označuje sa ako *Request Authenticator*, a ak o odpoveď značí sa ako *Response Authenticator*. Hodnota, v prípade že ide o Request Authenticator, je náhodné číslo v podobe MD5 súčtu. Ak sa v požiadavke nachádza parameter User-Password, tak je hodnota tohto parametru zašifrovaná skombinovaním autentifikátora a MD5 súčtu, vypočítaného zo zdieľaného hesla. Výsledok tejto operácie je skombinovaný pomocou funkcie XOR s užívateľským heslom. Výsledná hodnota je potom vložená do parametra User-Password. Preto je dôležité, aby bolo zdieľané heslo čo najviac bezpečné, pretože iba pomocou neho je zabezpečené samotné heslo užívateľa pri prenášaní sieťou.

Posledné pole, v ktorom sú obsiahnuté parametre jednotlivých typov paketov, má prízнačný názov **Attributes**. Tieto parametre sa označujú ako Attribute Value Pairs, skrátene AVP. Každý parameter pozostáva z troch polí. Prvé dva oktety v AVP sú **Type**, ktorý určuje o aký typ parametru ide (User-Name, User-Password, NAS-IP-Address,...) a **Length**, ktorý určuje celkovú veľkosť AVP. Posledné pole, označené ako **Value**, predstavuje hodnotu atribútu. Môže sa jednať o text, reťazec (string), adresu, celé číslo alebo čas. Text a reťazec môžu mať veľkosť až 253 oktetov, zatiaľ čo adresa, čas a číslo majú veľkosť 4 oktety. Pre IPv6 adresy boli vytvorené nové atribúty, u ktorých má adresa veľkosť 16 oktetov, teda 128 bitov. Tieto atribúty popisuje štandard RFC3162. Ďalším špecifickým typom atribútu je VSA, ktorý umožňuje výrobcom rozšíriť známe AVP o ich vlastné atribúty. [1, 40]

### 4.1.2 FreeRADIUS

FreeRADIUS patrí bezpochyby medzi najpopulárnejší software poskytujúci služby RADIUS servera. Je vyvíjaný ako open-source, vďaka čomu sa rýchlo vyvíja a má veľkú užívateľskú podporu. FreeRADIUS bol založený v roku 1999, pričom prvá verejná testovacia verzia vyšla v roku 2001 a odvtedy vychádza nová verzia každých pár mesiacov. Hoci je možné Free-

RADIUS využívať zdarma, poskytuje aj komerčnú podporu pod názvom Network RADIUS SARL. [42]

Popularita FreeRADIUSu môže byť pripísaná množstvu výhod, ktoré ma v porovnaní s inými implementáciami RADIUS servera. Medzi výhody môžme zaradiť tieto vlastnosti: [1, 41]

- **Open-source** - vďaka voľne dostupným kódom, si môže užívateľ prispôsobovať, meniť, rozširovať alebo opravovať funkcie FreeRADIUSu, podľa toho ako to sám uzná za vhodné. FreeRADIUS je šírený pod licenciou GNU GPL.
- **Bohatosť na funkcie** - má v sebe implementovanú širokú podporu typov autentifikácie, narozdiel od iných RADIUS serverov šírených ako open-source. Napríklad ako jediný z takýchto projektov podporuje protokol EAP. Taktiež má podporu virtuálnych serverov, pomocou ktorej môže jeden FreeRADIUS server spúšťať niekoľko na sebe nezávislých inštancii, pričom každej z nich môže byť priradená iná politika.
- **Modulárnosť** - FreeRADIUS poskytuje svoje funkcie v podobe zásuvných modulov. Funkcie, ktoré užívateľ nepoužíva, je možné jednoducho odstrániť z konfigurácie. Vypnutie nepotrebných modulov šetrí výkon servera, použitú pamäť a odstraňuje potencionálnu bezpečnostnú hrozbu. Pomocou modulov je možné integrovať podporu napríklad pre LDAP, SQL, PAM a mnohé ďalšie.
- **Škálovateľnosť** - na to aby FreeRADIUS prešiel od obsluhovania jednej požiadavky každých pár sekúnd, k obsluhe tisícov požiadaviek za sekundu, stačí iba zmeniť niekoľko parametrov z pôvodnej konfigurácie.
- **Rýchly vývoj** - vo FreeRADIUSe sa riadia mottom „release early, release often“ (vydávaj zavčasu, vydávaj často), vďaka čomu sú novinky v RADIUS protokole často implementované ako prvé práve do FreeRADIUSu.
- **Užívateľská komunita** - vďaka rozšírenosti FreeRADIUS servera má aj veľkú užívateľskú základňu, čo je veľmi užitočné pri riešení problémov, kedy je veľmi pravdepodobné, že podobný problém už vyriešil niekto pred nami.

Čas od času sa vo FreeRADIUSe, tak ako vo všetkých komplexnejších programoch, objaví nejaká ta zraniteľnosť, ktorá je ale vďaka aktívnemu vývoju relatívne rýchlo odstránená.

## 4.2 PAM moduly

PAM moduly predstavujú sadu knižníc v systéme Linux, ktoré umožňujú správcovi systému zvoliť, akým spôsobom budú jednotlivé aplikácie autentifikovať užívateľa. V minulosti sa pri aplikáciach vyžadujúcich autentifikáciu musel vybraný autentifikačný mechanizmus implementovať pri ich kompilácii. Aplikácie, ktoré využívajú PAM moduly, môžu využívať rôzne

spôsoby autentifikácie bez toho, aby sa muselo zasahovať priamo do nich. Každá takáto aplikácia má svoj konfiguračný súbor uložený v adresári `/etc/pam.d/`, v ktorom má definované, aké PAM moduly sa použijú na autentifikáciu. [43]

Flexibilita PAM modulov umožňuje zvoliť administrátorovi systému ľubovoľnú autentifikačnú schému, ktorá sa bude využívať v rámci celého systému alebo pri použití konkrétnej aplikácie. Pomocou PAM modulov tak môžme vytvoriť aj schému, pri ktorej bude na autentifikáciu použitých viacero PAM modulov za sebou, čím vytvoríme viacfaktorovú autentifikáciu. PAM modul môže na autentifikáciu využívať množstvo metód, medzi ktoré patrí autentifikácia užívateľským menom a heslom, tokenom (OTP, čipová karta,...) alebo biometriou (odtlačok prsta, rozpoznanie tváre,...).

Formát v akom sa konfigurujú jednotlivé PAM moduly v súbore `/etc/pam.conf` vyzerá takto: [43]

```
service type control module-path module-arguments
```

Pri konfigurácii aplikácii využívajúcich PAM moduly je syntax obsiahnutá v jednotlivých súboroch v adresári `/etc/pam.d/` rovnaká, s výnimkou absencie parametra *service*. V takom prípade slúži názov konkrétneho konfiguračného súboru namiesto tohto parametra.

Ďalšie je pole *type* a určuje, na aký typ úlohy bude daný modul slúžiť. Existujú štyri nasledujúce typy:

- *account* - slúži na vykonávanie operácii, ktoré nesúvisia priamo s autentifikáciou. Typicky sa používa na obmedzenie prístupu k službe v závislosti na dennom čase, prípadne v závislosti na aktuálne dostupných systémových prostriedkov (maximálny počet súčasne prihlásených užívateľov).

- *auth* - tento typ modulu poskytuje dve súčasti autentifikácie užívateľa. Pri prvej overí, či je užívateľ skutočne tým, za koho sa vydáva. Vykoná to vyzvaním užívateľa na zadanie mena a hesla alebo iných prostriedkov na identifikáciu. Druhá časť môže poskytnúť užívateľovi právo na používanie určitých služieb alebo napríklad mu udeliť členstvo v užívateľskej skupine.

- *password* - je vyžadovaný pri aktualizácii autentifikačného tokenu prideleného k užívateľovi (zmena hesla a pod.).

- *session* - má na starosti úkony, ktoré musia byť vykonané pred alebo po tom ako je užívateľovi pridelená služba. Zahŕňa to logovanie informácii týkajúcich sa napríklad výmeny dát s užívateľom, pripojenie priečinkov, atď.

Za definíciou akú úlohu bude zabezpečovať daný PAM modul, nasleduje parameter *control*, ktorým určíme akým spôsobom sa bude ďalej postupovať v prípade úspechu alebo zlyhania daného PAM modulu. Najčastejšie využívané hodnoty tohoto parametra sú:

- *required* - v prípade zlyhania takto označeného modulu, bude požiadavka napokon odmietnutá, ale až po tom ako sa použijú ostatné moduly rovnakého typu.
- *requisite* - podobný ako *required*, s tým rozdielom, že dôjde k odmietnutiu požiadavky ihneď po zlyhaní daného modulu, bez toho, aby sa zaoberal ďalšími modulmi v poradí.
- *sufficient* - ak modul s týmto parametrom uspeje (napr. s autentifikáciou) a žiadny predchádzajúci modul označený ako *required* nezlyhal, PAM odošle aplikácii informáciu o úspešnej autentifikácii, bez toho, aby sa zaoberal ďalšími modulmi v poradí. Ak dôjde k zlyhaniu modulu označeného ako *sufficient*, je tento modul ignorovaný a pokračuje sa s ďalšími modulmi v poradí.
- *optional* - úspech alebo zlyhanie takéhoto modulu je dôležité, iba v prípade, ak sa nachádza v konfiguračnom súbore ako jediný daného typu.

Predposledným parametrom je *module-path*, ktorý ako už názov napovedá, určuje cestu k danému PAM modulu. Môže tu byť zadaná celá cesta k súboru alebo iba názov PAM modulu, v prípade, že sa nachádza v defaultnom umiestnení PAM modulov. Umiestnenie pôvodných PAM modulov je v adresári `/lib/security/` alebo `/lib64/security/`, v závislosti na použitej architektúre. Toto umiestnenie taktiež závisí od použitej distribúcie Linuxu.

Posledný parameter je *module-arguments*, ktorý obsahuje argumenty jednotlivých PAM modulov. Tieto argumenty určujú špecifické správanie daného modulu, pričom každý modul má definované vlastné možné argumenty. [43]

Konfiguráciu PAM modulov by mal vykonávať iba skúsený užívateľ, pretože ich nesprávne nastavenie môže viesť k úplnému znemožneniu prístupu na zariadenie.

# 5 Praktická časť

Hlavným cieľom tejto práce je navrhnúť riešenie autentifikácie v prostredí IPv6 sietí s využitím biometrických metód. Ako autentifikačný server bude slúžiť virtuálny počítač s nainštalovaným softwarom FreeRADIUS, ktorý, ako aj názov napovedá, na autentifikáciu používa RADIUS protokol popísaný v kapitole 4.1. Tento server bude slúžiť zároveň aj ako MySQL databáza, do ktorej sa budú ukladať užívatelia, ktorý budú na prihlasovanie využívať RADIUS server. Ako RADIUS klient poslúži bežný desktopový počítač, a keďže sa má jednať o viacfaktorovú autentifikáciu s využitím biometrie, bude k nemu pripojená aj čítačka odtlačkov prsta. K tomu som ešte zvolil ako dodatočnú ochranu použiť mobilnú aplikáciu Google Authenticator, ku ktorej existuje PAM modul, a dá sa teda použiť pri autentifikácii na OS Linux.

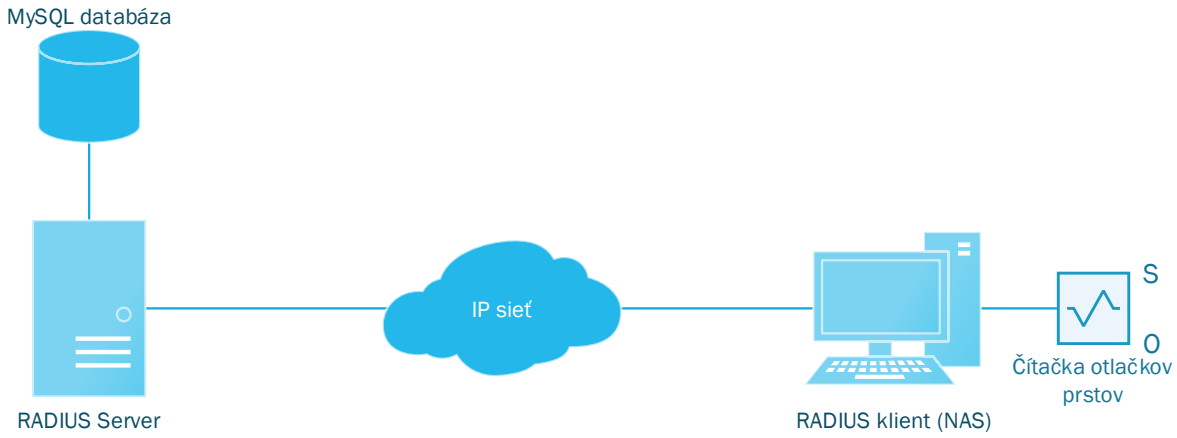
Server aj klient budú používať OS Ubuntu 14.04 LTS. Čo sa týka autentifikácie, tak tá bude prebiehať nasledovne:

1. Užívateľ pri prihlasovaní do systému bude vyzvaný na zadanie užívateľského mena a hesla. Tieto prihlasovacie údaje sa overia voči RADIUS serveru, a ten odpovie klientovi správou Access-Accept alebo Access-Reject.

2. V prípade, že prvá autentifikácia prebehne úspešne, bude užívateľ vyzvaný na zadanie vygenerovaného OTP z aplikácie Google Authenticator.

3. Ak obidve predchádzajúce autentifikačné metódy prebehnú korektne, užívateľ bude musieť pre úspešné prihlásenie ešte zosnímať svoj definovaný odtlačok prsta.

Takáto ochrana by mala spoľahlivo zabrániť neoprávnenému prihláseniu k danej užívateľskej stanici. Opísaný systém autentifikácie je zobrazený na obrázku 12.




Obr. 12: Schéma viacfaktorového autentifikačného prihlasovacieho systému.

## 5.1 Inštalácia a konfigurácia RADIUS servera

Keďže repozitáre Ubuntu majú v sebe obsiahnutú starú verziu programu FreeRADIUS s označením 2.1.12, ktorá je dnes už označovaná ako End Of Life, je potrebné novú verziu nainštalovať zo zdrojového kódu. Aktuálna verzia, označená ako stabilná, je 3.0.11 a je ju možné nainštalovať hneď dvomi spôsobmi. Pri prvom spôsobe sa zdrojové kódy skompilujú a nainštalujú pomocou série príkazov `configure`, `make` a `make install`. Nevýhodou pri takejto inštalácii je, že správca balíkov v Ubuntu neregistruje inštaláciu aplikácie, a preto nie je jednoduchá ani jej prípadná odinštalácia či aktualizácia. Druhou možnosťou je skompilovanie zdrojových kódov do inštalačných balíkov. Tento spôsob umožňuje jednoduchú inštaláciu, aktualizáciu, odinštaláciu alebo distribúciu softwaru. Všetky tieto operácie, až na distribúciu, sa vykonávajú pomocou správcu balíkov. Druhý spôsob je teda vhodnejší a preto použijem tento spôsob. [1]

Pred samotnou kompiláciou zdrojových kódov na inštalačné balíky, je potrebné doinštalovať niektoré nástroje nevyhnutné k správnemu vytvoreniu balíkov. Ako prvý nainštalujeme balíček `dpkg-dev` z repozitárov Ubuntu. Tento balíček poskytuje vývojárske nástroje potrebné na vytváranie inštalačných balíkov. Obsahuje v sebe množstvo nástrojov, vrátane kompilátora, spolu s programom `dpkg-buildpackage`, pomocou ktorého vykonáme samotné vytvorenie balíkov. [1]

Ako ďalší nainštalujeme program `fakeroot`. Následne môžeme pomocou príkazu `fakeroot` vytvoriť inštalačné balíky ako bežný užívateľ bez rootovských práv. Syntax tohoto príkazu je jednoduchá:

```
$ fakeroot <prikaz, ktorý chceme vykonat'>
```

Ako je možné vidieť, za kľúčovým slovom `fakeroot` nasleduje príkaz, ktorý chceme vykonať. Tento príkaz, ktorý vyvolá `fakeroot`, si bude myslieť, že bol spustený pod užívateľom root a má rootovské práva na manipuláciu so súbormi. Pri vytváraní inštalačných balíkov sa odporúča použiť príkaz `fakeroot` pred ich vytváraním pod rootom. [1]

Ako posledný nainštalujeme balík `ssl-cert`, bez ktorého by nebola možná kompilácia EAP modulu, a teda by sa nevytvorili všetky balíky potrebné k bezproblémovému chodu FreeRADIUSu. Všetky vyššie popísané balíky je možné nainštalovať z repozitárov spustením jedného príkazu:

```
$ sudo apt-get install dpkg-dev fakeroot ssl-cert
```

Teraz sa môžeme pustiť do inštalácie samotného FreeRADIUS servera. Najprv je potrebné stiahnuť zdrojové kódy tohto RADIUS servera. Keďže web stránky FreeRADIUSu fungujú iba pod IPv4 a virtuálny server, na ktorom bude prebiehať inštalácia, funguje iba pod IPv6, musel som potrebné súbory stiahnuť na PC, ktoré malo aj IPv4 adresy. Stiahnutý súbor som následne skopíroval do priečinka */home/student* na virtuálnom serveri pomocou programu `scp`. Vykonal som to pomocou nasledujúcej série príkazov:

```
$ wget ftp://ftp.freeradius.org/pub/freeradius/freeradius-server
   -3.0.11.tar.gz
$ scp -6 freeradius-server-3.0.11.tar.gz student@[2001:718:1001:2c6
   ::211]:~/
```

Na serveri teraz môžme rozbaliť stiahnutý TAR archív a zmeniť aktuálny pracovný adresár na novovytvorený, ktorý vznikol po rozbalení:

```
$ tar zxfv freeradius-server-3.0.11.tar.gz
$ cd freeradius-server-3.0.11
```

Na to aby sme mohli skompilovať zdrojové kódy a nainštalovať FreeRADIUS, je potrebné ešte doinštalovať mnoho takzvaných závislostí. Správca balíkov `dpkg` má v sebe obsiahnutý nástroj na kontrolu závislostí, ktoré sú potrebné na kompiláciu a správnu funkčnosť balíkov. Ide o nástroj `dpkg-checkbuilddeps`, ktorý spustíme v adresári so zdrojovými kódmi:

```
$ dpkg-checkbuilddeps
```

Výstup tohoto príkazu je výpis balíkov, ktoré je potrebné doinštalovať z repozitárov pred samotnou kompiláciou inštalačných balíkov FreeRADIUSu. Zvyčajne sa tu nachádza pomerne veľké množstvo závislostí. Ak sa vo výpise medzi dvomi balíkmi nachádza symbol „|“, je potrebné nainštalovať iba jeden z nich. Vybrané závislosti v mojom prípade potom nainštalujem spustením príkazu:

```
$ sudo apt-get install debhelper quilt autotools-dev libcurl4-openssl
   -dev libcap-dev libgdbm-dev libiodbc2-dev libjson0-dev libkrb5-dev
    libldap2-dev libpam0g-dev libpcap-dev libperl-dev libmysqlclient-
   dev libpq-dev libreadline-dev libsasl2-dev libsqlite3-dev libssl-
   dev libtalloc-dev libwbclient-dev libyubikey-dev libykclient-dev
   libmemcached-dev libhiredis-dev python-dev samba-dev
```

Po dokončení inštalácie týchto balíkov, môžme overiť, že už skutočne nie je potrebné doinštalovať d'alšie závislostí. Vykonáva sa to opätovným spustením príkazu `dpkg-checkbuild deps`, ktorého výstup by mal byť, v prípade korektnej inštalácie závislostí, prázdny. V prípade, že sa vo výpise objavia nejaké balíky, je potrebné ich doinštalovať.

Ak overenie prebehlo v poriadku, a teda bol výstup prázdny, môžme prejsť k samotnej kompilácii zdrojových súborov a vytvoreniu inštalačných balíkov. Tento proces je pomerne zdĺhavý a jeho dĺžka závisí od výpočtového výkonu PC, na ktorom je spustený. Pri tomto kroku využijeme program `fakeroot` a vytvorenie balíkov spustíme pomocoou príkazu `dpkg-build package`. Za týmto príkazom použijeme prepínač `-b`, ktorým určíme, že sa majú vytvoriť iba binárne súbory, a prepínač `-uc` na preskočenie podpisu pomocou GPG kľúča. Výsledný príkaz bude teda vyzerať nasledovne: [1]

```
$ fakeroot dpkg-buildpackage -b -uc
```

Ak sa pri kompilácii nevyskytli žiadne chyby, novovytvorené inštalačné balíky by sa mali nachádzať o priečinok vyššie. Vytvorené balíky si môžme vypísať pomocou príkazu `ls`, ktorý presmerujeme na príkaz `grep`, ktorý nám vyfiltruje iba inštalačné balíčky nachádzajúce sa v danom adresári. Výstup takého príkazu je zobrazený na obrázku 13 a samotný príkaz bude vyzerať takto:

```
$ ls ../ | grep .deb
```

```
student@radius:~/freeradius-server-3.0.11$ ls ../ | grep .deb
freeradius_3.0.11+git_amd64.deb
freeradius-common_3.0.11+git_all.deb
freeradius-config_3.0.11+git_amd64.deb
freeradius-dbg_3.0.11+git_amd64.deb
freeradius-dhcp_3.0.11+git_amd64.deb
freeradius-iodbc_3.0.11+git_amd64.deb
freeradius-krb5_3.0.11+git_amd64.deb
freeradius-ldap_3.0.11+git_amd64.deb
freeradius-memcached_3.0.11+git_amd64.deb
freeradius-mysql_3.0.11+git_amd64.deb
freeradius-postgresql_3.0.11+git_amd64.deb
freeradius-redis_3.0.11+git_amd64.deb
freeradius-rest_3.0.11+git_amd64.deb
freeradius-utils_3.0.11+git_amd64.deb
freeradius-yubikey_3.0.11+git_amd64.deb
libfreeradius3_3.0.11+git_amd64.deb
libfreeradius-dev_3.0.11+git_amd64.deb
student@radius:~/freeradius-server-3.0.11$
```

Obr. 13: Zobrazenie skompilovaných inštalačných balíkov.

Ako je možné vidieť na obrázku 13, vytvorených balíkov je viacero. Na základnú funkciu FreeRADIUS servera postačuje však nainštalovať iba niektoré z nich, pričom jeden na druhom závisia. Pre pohodlnejšiu inštaláciu sa presunieme o adresár vyššie a pomocou príkazu `dpkg -i`, kde prepínačom `-i` určujeme, že vybrané balíky sa majú nainštalovať do systému, nainštalujeme potrebný základ pre FreeRADIUS:

```
$ cd ../
$ sudo dpkg -i freeradius_3.0.11+git_amd64.deb freeradius-common_3
   .0.11+git_all.deb freeradius-config_3.0.11+git_amd64.deb
   freeradius-utils_3.0.11+git_amd64.deb libfreeradius3_3.0.11+
   git_amd64.deb
```

V prípade, ak budeme potrebovať využívať ďalšie súčasti FreeRADIUS servera, jednoducho podobným spôsobom doinštalujeme balíky, ktoré potrebujeme. Prípadne ich môžme nainštalovať všetky naraz. Mohli by sme spustiť inštaláciu vypísaním všetkých balíkov, ale bolo by to príliš zdĺhavé. Namiesto toho môžme využiť regulárnych výrazov a spustiť inštaláciu jedným krátkym príkazom:

```
$ sudo dpkg -i *freeradius*_3.0.10+*.deb
```

Tým sa dokončí inštalácia FreeRADIUS servera a spustí ho ako službu na pozadí. Či je FreeRADIUS naozaj spustený a počúva na svojich defaultných portoch, môžme overiť programom `netstat`, ktorý slúži na zisťovanie sieťových štatistík. Nás bude zaujímať, ktoré porty na serveri sú otvorené. Za príkazom `netstat` použijeme prepínač `-6unlp`, kde číslo „6“ označuje použitý protokol IPv6, „u“ označuje UDP protokol, písmenom „l“ nastavíme aby sa zobrazovali porty, na ktorých server počúva a pomocou „p“ zobrazíme PID procesu a jeho názov. Tento príkaz následne presmerujeme na nástroj `grep`, ktorým vyfiltrujeme iba procesy s názvom *freeradius*. Príkaz a jeho predpokladaný výstup vyzerajú následovne:

```
$ sudo netstat -6unlp | grep freeradius
udp6    0     0 :::1812     :::*     54750/freeradius
udp6    0     0 :::1813     :::*     54750/freeradius
udp6    0     0 :::33957    :::*     54750/freeradius
```

Pre nás je dôležitý prvý riadok výstupu tohto príkazu, pretože na porte 1812 prebieha autentifikácia na RADIUS serveri. Funkčnosť autentifikácie je možné overiť pomocou nástoja `radtest`, ktorý patrí pod balík `freeradius-utils`. Inštalácia tohoto balíka prebehla vyššie a nie je ho teda potrebné doinštalovať. Najprv je ale potrebné v konfiguračnom súbore `users`, ktorý sa nachádza v adresári `/etc/freeradius/`, pridať nejakého užívateľa. Na testovanie nám poslúži predvytvorený užívateľ v tomto súbore s menom *bob*, ktorý je v pôvodnom nastavení iba zakomentovaný a nachádza sa na riadku 87. Odkomentujeme tohoto užívateľa a reštartujeme FreeRADIUS. To vykonáme príkazmi a úpravou konfiguračného súboru následovne:

```
$ sudo nano /etc/freeradius/users
...
bob   Cleartext-Password := "hello"
      Reply-Message := "Hello, \%{User-Name}"
...
$ sudo service freeradius restart
```

Keby sme teraz spustili program `radtest`, skončil by následujúcou chybou:

```
(0) Error parsing "stdin": Failed resolving "ubuntu" to IPv6 address:
    Temporary failure in name resolution.
```

Je to spôsobené tým, že RADIUS pri autentifikácii prekladá doménové meno NAS, ktorý je v tomto prípade zároveň aj RADIUS server, a má ho nastavené na *ubuntu*. Ak sa mu doménové meno nepodarí preložiť na IP adresu, `radtest` ani neodošle žiadosť o autentifikáciu na RADIUS server. Pre účely otestovania funkčnosti autentifikácie RADIUS servera, postačí pridať v súbore `/etc/hosts` na riadok, na ktorom sa nachádza IPv6 localhost adresa, doménové meno počítača, ktoré je v tomto prípade *ubuntu*. Tento riadok by mal potom vyzerať takto:

```
::1     ip6-localhost ip6-loopback ubuntu
```

Keď je tento súbor upravený, môžme postúpiť k otestovaniu autentifikácie pomocou príkazu `radtest`. Tento príkaz ma nasledujúcu syntax:

```
radtest [OPTIONS] user passwd radius-server[:port] nas-port-number
    secret
```

Na miesto OPTIONS vkladáme parametre s ktorými chceme program spustiť. V našom prípade použijeme iba paramter `-6`, ktorým definujeme aby sa na autentifikáciu použil protokol IPv6. Používateľské meno vkladáme namiesto `user` a heslo tohto užívateľa dávame namiesto `passwd`. Za nimi nasleduje IP adresa alebo doménové meno RADIUS servera nasledované portom, na ktorom RADIUS počúva pre autentifikáciu. Ďalej nasleduje `nas-port-number`, čo je hodnota atribútu NAS-Port v požiadavke Access-Request odoslanej na RADIUS server. Tento parameter môže mať hodnotu ľubovoľného celého čísla v rozmedzí od 0 až $2^{31}$. Ako posledné je potrebné zadať heslo, ktorým sa NAS autentifikuje voči RADIUS serveru. V pôvodnom nastavení FreeRADIUSu je pre klienta localhost prednastavené heslo testing123. V prípade, že by sa nadefinované heslo na serveri nezhodovalo s heslom, ktorým sa autentifikuje samotný NAS, server zahodí paket a požiadavkou sa d'alej nijako nezaoberá. V našom prípade bude príkaz vyzerať takto:

```
$ radtest -6 bob hello ip6-localhost:1812 10 testing123
```

Ak všetká konfigurácia prebehla korektne, server by mal odpovedať správou Access-Accept, ktorou potvrdzuje, že autentifikácia zadanej kombinácie mena a hesla, sa úspešne autentifikovala voči RADIUS serveru. Výstup z vyššie uvedeného príkazu zobrazuje obrázok 14.

```
student@ubuntu:~$ radtest -6 bob hello ip6-localhost:1812 0 testing123
Sent Access-Request Id 30 from [::]:52383 to [::1]:1812 length 85
        User-Name = "bob"
        User-Password = "hello"
        NAS-IPv6-Address = ::1
        NAS-Port = 0
        Message-Authenticator = 0x00
        Cleartext-Password = "hello"
Received Access-Accept Id 30 from [::1]:1812 to [::]:0 length 32
        Reply-Message = "Hello, bob"
student@ubuntu:~$
```

Obr. 14: Výstup úspešnej autentifikácie pomocou utility `radtest`.

RADIUS server je spojený s využívaním doménových mien, keď pri posielaní požiadavku prekladá doménové meno klienta na IP adresu. Tieto doménové mená musia byť niekde nadefinované. Úprava súboru `/etc/hosts` by bola pri väčšom počte klientských staníc zdĺhavá a nepraktická. Navyše pri akejkoľvek zmene, by bolo potrebné upravovať súbor `hosts` na každom zariadení zvlášť. Preto je na mieste použiť DNS server, ktorý odstraňuje všetky tieto nedostatky. DNS server sa bude nachádzať na tom istom virtuálnom serveri ako aj FreeRADIUS a jeho konfigurácia bude popísaná v následujúcej podkapitole.

### 5.1.1 Inštalácia a konfigurácia DNS servera

DNS server nám bude slúžiť na preklad doménových mien koncových staníc na IP adresy. Keďže je práca venovaná IPv6 sieťam, popíšem nastavenie DNS servera iba na preklad IPv6 adries. Pre Linux existuje niekoľko programov, ktoré slúžia ako DNS server. Medzi najpoužívanejšie patrí `BIND9`. Na jeho inštaláciu využijeme repozitáre Ubuntu a nainštalujeme ho pomocou príkazu:

```
$ sudo apt-get install bind9
```

Po nainštalovaní môžeme začať s konfiguráciou. Pre testovacie účely som si vytvoril doménu s názvom *raddp.cz*. Všetky potrebné konfiguračné súbory sa nachádzajú v adresári `/etc/bind/`. Keďže sú tieto súbory obsiahlejšie, ich výsledný obsah som vložil do práce ako prílohu B. V konfiguračnom súbore `named.conf.local` určíme, že tento DNS server bude typu *Master* pre doménu *raddp.cz* a definujeme DNS zóny, o ktoré sa bude starať. Prvá zóna bude dopredná a bude slúžiť na preklad doménového mena na IP adresu. Druhá bude takzvaná reverzná zóna a tá bude zasa naopak prekladať IP adresy na doménové mená. Výsledná podoba súboru sa nachádza v prílohe B.1. [8, 9]

Názov reverznej domény je zámerne taký dlhý, aby sa v súbore s reverznými záznamami nenachádzali vždy celé IPv6 adresy pri každom zázname, ale iba tá časť, ktorá sa odlišuje. Reverzné záznamy totiž nepodporujú skrátený formát IPv6 adries s vynechanými nulami, ale je potrebné ich vypisovať celé oddelené bodkami. V mojom prípade je IP adresa virtuálneho servera `2001:718:1001:2c6::211` a IP adresa počítača `2001:718:1001:2c8:76d4:35ff:fe7c:ece`. Ako je možné vidieť, časť `2001:718:1001:2c` majú spoločnú, takže je možné vytvoriť zónu, ktorá bude spravovať všetky adresy, ktoré majú takýto prefix. [9]

Teraz môžme pristúpiť na vytvorenie samotných súborov pre zóny nadefinované v súbore `named.conf.local`. Ako prvú vytvoríme zónu s doprednými záznamami. Na vytvorenie našej zóny môžme využiť vzorový súbor `db.empty`, ktorý obsahuje iba základné parametre. Tento súbor skopírujeme a upravíme ho podľa našich potrieb. V mojom prípade som postupoval takto:

```
$ sudo cp /etc/bind/db.empty /etc/bind/db.raddp.cz
$ sudo nano /etc/bind/db.raddp.cz
```

Konečný obsah upraveného súboru sa nachádza v prílohe B.2.

Pre RADIUS server som zvolil doménové meno `radius.raddp.cz` a pre klientskú stanicu `client.raddp.cz`. Je pravdepodobné, že pri nasadzovaní DNS serveru v reálnom prostredí, by bolo klientských staníc určite viac, a bolo by vhodné zvoliť iný systém pomenovania staníc. V tomto konfiguračnom súbore reprezentuje symbol „@“ samotnú doménu `raddp.cz`. V prípade ak sa do súboru pre zónu pridávajú nové alebo upravujú stávajúce záznamy, vždy je potrebné navýšiť hodnotu parametra `Serial`, inak by `BIND9` nebral dané úpravy do úvahy.

Ako je možné vidieť v konfiguračných súboroch, boli tu použité záznamy NS a AAAA. Okrem nich existujú aj ďalšie záznamy, z ktorých tie najpoužívanejšie sú následujúce: [8, 10]

- **A záznam** - slúži na mapovanie IP adresy na doménové meno.
- **AAAA záznam** - zastáva rovnakú funkciu ako A záznam, s tým rozdielom, že definuje doménové meno pre IPv6 adresu.
- **NS záznam** - určuje, ktorý server obsluhuje danú zónu a musí odkazovať na A záznam. Najčastejšie sa vyskytujú dva NS záznamy, pričom prvý z nich určuje primárny server a druhý sekundárny server, ktorý slúži ako záloha v prípade výpadku primárneho servera.
- **CNAME záznam** - používa sa na vytvorenie tzv. aliasu pre existujúce doménové meno, prípadne A alebo AAAA záznam.
- **MX záznam** - určuje, na aký mailový server sa má poslať email smerujúci na danú doménu. Môže existovať viacero MX záznamov, pričom každému sa prideľuje priorita. Ak nie je možné doručiť mail pomocou záznamu s najvyššou prioritou, postupne sa prechádzajú záznamy s nižšími prioritami.
- **PTR záznam** - používa sa v reverznej zóne a slúži na preklad IP adresy na doménové meno. Pre každý A a AAAA záznam v doprednej zóne, ktorý odkazuje na odlišnú IP adresu, musí existovať PTR záznam.

Ďalej je potrebné vytvoriť súbor s reverznou zónou. Opäť môžeme využiť vzorový súbor `db.empty`, ktorý skopírujeme a premenujeme na názov reverznej zóny definovaný v súbore `named.conf.local`:

```
$ sudo cp /etc/bind/db.empty /etc/bind/db.2001.718.1001.2c
$ sudo nano /etc/bind/db.2001.718.1001.2c
```

Výsledný súbor s reverznými záznamami je uvedený v prílohe B.3.

Aby sme mohli prekladať doménové mená aj mimo našu doménu, je potrebné nastaviť takzvaný forwardovací server, ktorý bude tieto požiadavky obsluhovať. Keďže testovanie navrhnutého systému prebieha v počítačovej sieti VŠB-TUO, ako forwardovací server je možné použiť iba oficiálny server s IP adresou `2001:718:1001::53`, pretože ostatné sú blokované. Forwardovací server sa nastavuje v súbore `named.conf.options`.V tomto súbore môže byť definovaných aj viacero serverov, ktoré by sa použili v prípade, že by prvý nebol schopný preložiť doménove meno. V mojom prípade však bude len jeden. Takže otvoríme konfiguračný súbor a pridáme do neho tento DNS server: [8]

```
options {
      ...
      forwarders {
```

```
            2001:718:1001::53;
        };
        ...
}
```

Teraz by sme mohli reštartovať službu `bind9` a dúfať, že sme pri konfigurácii nespravili žiadnu chybu. Lepším riešením je však použiť utilitu `named-checkzone`, ktorá je súčasťou balíka `bind9`. Pomocou tejto utility môžme skontrolovať správnosť konfiguračných súborov jednotlivých zón a umožňuje nám tak vyhnúť sa prípadným problémom, ktoré by spôsobila nesprávna konfigurácia. Tento nástroj má nasledujúci syntax: [11]

```
$ named-checkzone <názov zóny> <názov súboru>
```

Názov zóny a názov súboru so zónou udávame tak, ako sme ich definovali v konfiguračnom súbore `named.conf.local`. Výstupom tohoto príkazu je buď nejaká chybová hláška, ktorá presne udáva o aký typ nesprávnej konfigurácie ide a kde sa nachádza, alebo správa o načítaní zóny so sériovým číslom, nasledovanou správou `OK`. Príklad výstupu so správnou konfiguráciou je zobrazený na obrázku 15. Ak bola naša konfigurácia správna, môžme pristúpiť k reštartovaniu služby `bind9` pomocou príkazu:

```
$ sudo service bind9 restart
```

```
student@ubuntu:~$ named-checkzone raddp.cz /etc/bind/db.raddp.cz
zone raddp.cz/IN: loaded serial 1
OK
student@ubuntu:~$
student@ubuntu:~$ named-checkzone c.2.0.1.0.0.1.8.1.7.0.1.0.0.2.ip6.arpa /etc/bi
nd/db.2001.718.1001.2c
zone c.2.0.1.0.0.1.8.1.7.0.1.0.0.2.ip6.arpa/IN: loaded serial 1
OK
student@ubuntu:~$
```

Obr. 15: Výstup úspešného overenia konfigurácie zón pomocou `named-checkzone`.

Teraz je potrebné nastaviť na serveri a aj na klientskej stanici, aby na preklad doménových mien používal nami vytvorený DNS server. Na strane servera je konfigurácia o niečo odlišnejšia ako na strane klienta. Je to spôsobené tým, že na strane servera je zvyčajne nakonfigurovaná statická IP adresa, kdežto u klienta býva pridelená pomocou DHCP servera. Najprv popíšem nastavenie na serveri.

Nastavenie DNS servera sa vykonáva úpravou súboru `/etc/resolv.conf`, ktorý ale dynamicky spravuje program s názvom `resolvconf`. Keby sme upravili tento konfiguračný súbor ručne, nastavenie by vydržalo do najbližšieho reštartu servera, prípadne do vypnutia a opätovného zapnutia sieťového rozhrania. Program `bind9` je zodpovedný za pridanie tzv. loopback adresy do súboru `resolv.conf`. V defaultnom nastavení je však táto funkcia vypnutá a je potrebné ju zapnúť úpravou konfiguračného súboru `bind9`, ktorý sa nachádza v adresári `/etc/default/`. Tento súbor otvoríme a upravíme v ňom riadok `RESOLVCONF=no` na

`RESOLVCONF=yes`. Aby `bind9` upravil súbor `resolv.conf`, je ho potrebné reštartovať. Vykonáme to následujúcimi príkazmi a úpravou tohto súboru: [11]

```
$ sudo nano /etc/default/bind9
...
RESOLVCONF=yes
...
$ sudo service bind9 restart
```

Ďalšia vec, ktorú je vhodné pridať do konfiguračného súboru `resolv.conf`, je prehľadávanie našej domény. Nastavuje sa to pomocou pridania riadku `search raddp.cz` do tohto súboru. Opäť je tam však pridávaný dynamicky, a je ho preto treba nastaviť v konfiguračnom súbore `/etc/network/interfaces`, ktorý slúži na nastavenie sieťových adaptérov. Funkcia `search` sa využije v prípade, že budeme chcieť preložiť iba hostname bez zadania domény. Napríklad nebudeme musieť zadávať celé doménové meno `radius.raddp.cz`, ale bude stačiť ak použijeme iba názov `radius`. Do súboru `interfaces` je potrebné pridať za nastavenie loopback rozhrania riadok `dns-search raddp.cz`. Po úprave tohto súboru je potrebné rozhranie loopback vypnúť a znova zapnúť, aby nastavenie bolo účinné. Použitie tohto rozhrania má výhodu v tom, že nie je potrebné reštartovať celý server, ako by to bolo nutné v prípade keby sme tento riadok pridali pod fyzické rozhranie. Vzhľadom na to, že sa na mnou využívaný virtuálny server pripájam iba pomocou SSH, nesprávna konfigurácia fyzického rozhrania, by spôsobila nedostupnosť servera. Popísané kroky sa vykonajú pomocou príkazov a úpravy súboru následovne:

```
$ sudo nano /etc/network/interfaces
...
auto lo
iface lo inet loopback
      dns-search raddp.cz
...
$ sudo ifdown lo
$ sudo ifup lo
```

Ak všetky nastavenia prebehli korektne mal by súbor `resolv.conf` vyzerať tak ako na obrázku 16. Hoci je tu definovaná iba IPv4 adresa rozhrania loopback, na funkčnosť prekladu doménových mien na IPv6 adresy to nemá vplyv.

```
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 127.0.0.1
search raddp.cz
```

Obr. 16: Ukážka konfiguračného súboru `resolv.conf`.

Na klientskej strane býva vo väčšine prípadov prideľovaná IP adresa, spolu s adresou DNS servera, pomocou DHCP. V Ubuntu je integrovaný nástroj s názvom Network Manager, ktorý slúži na automatickú konfiguráciu sieťových rozhraní, s čo najmenšou potrebou zásahu od užívateľa. Ak teda nastavíme nejaký parameter (napr. IP adresu) staticky bez využitia Network Managera, tak ten nám ju bude v pravidelných intervaloch premazávať a nastavovať si parametre, ktoré má definované vo svojich konfiguračných súboroch. Keďže v mojom prípade nemám možnosť prevádzkovať DHCP server v sieti VŠB-TUO, musel som vyriešiť ako nastaviť používanie DNS servera na klientských staniciach bez toho, aby mi ich neustále prepisoval Network Manager. To by nastalo v prípade, že by som v konfiguračnom súbore `resolv.conf` ručne nastavil používanie mnou vytvoreného DNS. [12]

Je teda potrebné upraviť konfiguračné súbory samotného Network Managera. Tie sa nachádzajú v adresáry `/etc/NetworkManager/system-connections/`. Pre každé sieťové rozhranie sa tu nachádza samostatný konfiguračný súbor. V mojom prípade budem upravovať súbor s názvom `Wired connection 1`. Je potrebné upraviť sekciu pre IPv6, kde definujeme náš DNS server, aká doména sa ma prehľadávať a nastavíme aby pri získavaní adresy z DHCP ignoroval prideľovaný DNS server. Všetky potrebné úkonu vykonáme následujúcimi príkazmi a úpravou súboru: [12]

```
$ sudo nano /etc/NetworkManager/system-connections/Wired\ connection\
    1
...
[ipv6]
method=auto
dns=2001:718:1001:2c6::211;
dns-search=raddp.cz;
ignore-auto-dns=true
...
```

V tejto chvíli môžme reštartovať Network Managera, aby sa dané zmeny aplikovali. Keby sme však nahliadli do súboru `resolv.conf`, bola by v ňom zobrazená adresa DNS servera `127.0.1.1`. Je to spôsobené tým, že Network Manager využíva na preklad doménových mien program s názvom `dnsmasq`. Jedná sa o jednoduchý forwardovací DNS server, ktorý je implementovaný takmer v každej distribúcii Linuxu. Tento program ukladá do vyrovnávacej pamäte DNS záznamy, aby pri opakovanej požiadavke na preklad toho istého doménového mena, nebolo potrebné sa znova dotazovať vzdialeného servera, čo by malo urýchliť preklad často sa opakovaných mien. [13]

V prípade, že nechceme tento program používať, stačí v konfiguračnom súbore Network Managera „zakomentovať“ riadok, ktorý zapína použitie `dnsmasq`. Následne je ešte potrebné reštartovať Network Managera a v súbore `resolv.conf`, by sa mala objaviť nami nastavená IP adresa pre DNS server. Vykonáme to týmito príkazmi a úpravou súboru:

```
sudo nano /etc/NetworkManager/NetworkManager.conf
...
[main]
plugins=ifupdown,keyfile,ofono
#dns=dnsmasq
...
$ sudo service network-manager restart
```

Keď už máme nastavený správny DNS server, môžeme overiť, či korektne prekladá nami nadefinované doménové mená. Existuje viacero spôsobov to overiť. Jedným z nich je nástroj `nslookup`. Pomocou tohto programu, môžeme overiť preklad doménového mena na IP adresu, a taktiež môžme overiť správnu konfiguráciu reverzného záznamu. Pri kontrole prekladu na IPv6 adresy, je potrebné za príkazom `nslookup` použiť prepínač `-querytype=AAAA`, ktorým definujeme, že sa pýtame na záznam AAAA. Bez tohoto prepínača by použil svoje predvolené nastavenie, čo je dotaz na záznam A, ktorý nemáme definovaný a skončil by teda chybou. Keďže sme definovali, že má prehľadávať doménu `raddp.cz`, nie je potrebné písať do tohoto príkazu celé doménové meno, ale stačí iba názov pred `raddp.cz`. Príkaz na otestovanie funkčnosti prekladu doménového `radius.raddp.cz` bude vyzerať takto: [14]

```
$ nslookup -querytype=AAAA client
```

Pri testovaní správnosti konfigurácie reverzného záznamu, stačí za príkaz `nslookup` dopísať IP adresu a program sa ju pokúsi preložiť pomocou nášho DNS servera na doménové meno. Ak je naša konfigurácia správna, výstupom nástroja `nslookup` bude použitý server na preklad a IP adresa alebo doménové meno príslušiace k danému dotazu. Ukážkové výstupy testovanie môjho DNS servera sú zobrazené na obrázku 17.

```
student@ubuntu:~$ nslookup -querytype=AAAA client
Server:         127.0.0.1
Address:        127.0.0.1#53

client.raddp.cz has AAAA address 2001:718:1001:2c8:76d4:35ff:fe7c:ece

student@ubuntu:~$
student@ubuntu:~$ nslookup 2001:718:1001:2c8:76d4:35ff:fe7c:ece
Server:         127.0.0.1
Address:        127.0.0.1#53

e.c.e.0.c.7.e.f.f.f.5.3.4.d.6.7.8.c.2.0.1.0.0.1.8.1.7.0.1.0.0.2.ip6.arpa      n
ame = client.raddp.cz.

student@ubuntu:~$
```

Obr. 17: Testovanie správnej funkčnosti DNS servera pomocou nástroja `nslookup`.

Keď máme overenú správnu funkčnosť DNS servera, môžme na serveri a jednotlivých staniciach nastaviť ich doménové meno. V mojom prípade to bude na strane servera meno

*radius.raddp.cz* a na strane klienta *client.raddp.cz*. Na konfiguráciu doménového mena slúži nástroj `hostnamectl`. Tento nástroj vkladá nami zadané doménové meno do konfiguračného súboru `/etc/hostname` a zároveň ho ihneď aplikuje. Keby sme ručne upravovali tento súbor, bolo by potrebné reštartovať PC, čo nie je vždy žiadúce. Pre nastavenie doménového mena *radius.raddp.cz* použijeme príkaz: [15]

```
$ sudo hostnamectl set-hostname radius.raddp.cz
```

### 5.1.2 Konfigurácia klienta

Na to aby sme mohli overovať užívateľov pomocou NAS servera, ktorý je v našom prípade klientská stanica, je potrebné tento NAS server najprv autentifikovať. FreeRADIUS server bude akceptovať požiadavky na autentifikáciu iba od NAS definovaných vo svojom konfiguračnom súbore. Konkrétne ide o súbor `/etc/freeradius/clients.conf`, v ktorom sa nachádzajú všetci takýto klienti. Ak by v danom súbore nebola definovaná IP adresa, prípadne doménové meno, alebo by sa definované heslo nezhodovalo s tým od NAS, RADIUS server by požiadavku na autentifikáciu vždy zamietol, a to aj v prípade, že by prihlasovacie údaje boli správne.

Na základnú konfiguráciu je potrebné do spomenutého konfiguračného súboru vložiť niekoľko riadkov s následujúcou syntaxou:

```
client NÁZOV {
        ipaddr    = ADRESA
        secret    = HESLO
}
```

Vo verzii FreeRADIUS 1.x bolo potrebné aby `NÁZOV` bola IP adresa klienta. Od verzie 2.0 to môže byť ľubovoľný reťazec a IP adresa klienta sa vkladá do poľa `ipaddr`. Na mieste reťazca `ADRESA`, sa teda môže nachádzať IPv4 alebo IPv6 adresa. Prípadne tu môžme vložiť aj doménové meno, pričom ak existujú na toto doménové meno záznamy A aj AAAA, bude uprednostnený záznam A. Namiesto kľúčového slova `ipaddr` sa môžu použiť aj `ipv4addr` alebo `ipv6addr`. Ako ich názov napovedá, prvý slúži len pre IPv4 adresy a druhý len pre IPv6 adresy. Nemusí sa však jednať iba o unikátnu IP adresu, ale môže sa do tohoto poľa vložiť aj celý subnet, v tvare napríklad 192.168.1.0/24.

Ďalším dôležitým parametrom, ktorý sa tu udáva je `secret`. Nastavuje sa ním heslo, ktoré je použité na šifrovanie hesla užívateľa a podpis paketov vymieňaných medzi NAS a FreeRADIUSom. Bezpečnosť RADIUS protokolu je veľmi závislá práve na tomto hesle. Preto sa v reálnom prostredí odporúča používať heslo zložené z veľkých a malých písmen v kombinácii s číslami. Malo by mať minimálne 8 znakov a malo by byť náhodné. Nemali by sa používať bežné slovné spojenia alebo čokoľvek iné, čo sa dá ľahko rozpoznať.

V mojom prípade do konfiguračného súboru `clients.conf`, ako názov klientov vložím `private _ipv6 _subnet`, pretože tu definujem ako klientov celý subnet `2001:718:1001:`

2c8::/64, ktorý je pridelený pre laboratórium. Heslo som nastavil na testing123, ktoré by v reálnych podmienkach bezpečné nebolo, no na moje testovacie účely je dostačujúce. Výsledný súbor otvorím a na jeho koniec vložím tieto následujúce riadky:

```
$ sudo nano /etc/freeradius/clients.conf
...
client private_ipv6_subnet {
      ipv6addr = 2001:718:1001:2c8::/64
      secret   = testing123
}
```

Aby sme mohli na koncovej stanici overiť správnosť nastavenia, je potrebné nainštalovať balík freeradius-utils, ktorý obsahuje nástroj radtest. K inštalácii potrebného nástroja radtest, ktorý slúži na overenie funkčnosti autentifikácie, môžme využiť nami vytvorené balíky z inštalácie FreeRADIUS servera. Pre nás je dôležitý už spomenutý inštalačný balík freeradius-utils. Tento balík je však ešte závislý na balíkoch freeradius-common, freeradius-config a libfreeradius3. Skopírujeme teda všetky tieto balík zo servera na klienta pomocou programu scp: [16]

```
scp student@[radius.raddp.cz]:~/\{freeradius-common_3.0.11+git_all.
   deb,freeradius-config_3.0.11+git_amd64.deb,freeradius-utils_3
   .0.11+git_amd64.deb,libfreeradius3_3.0.11+git_amd64.deb\} ~/
```

Keby sme chceli teraz tieto skopírované balíky nainštalovať, inštalácia by sa skončila chybou, pretože tieto balíky sú závisle ešte od balíka libtalloc2, ktorý doinštalujeme z repozitárov Ubuntu. Vzápätí môžme nainštalovať potrebné balíky a vykonáme to touto sériou príkazov:

```
$ sudo apt-get install libtalloc2
$ sudo dpkg -i *freeradius*_3.0.11+*.deb
```

Ak všetko prebehlo v poriadku, môžme pomocou nástroja radtest overiť správnu konfiguráciu NAS klienta na FreeRADIUS serveri. Opäť využijeme užívateľa s menom *bob*. Použijeme na to tento príkaz:

```
$ radtest -6 bob hello radius.raddp.cz 0 testing123
```

Ak bola všetká konfigurácia správna, mali by sme dostať odpoveď Access-Accept. Výstup nástroja radtest by mal byť teda rovnaký ako na obrázku 14, s výnimkou zdrojovej a cieľovej adresy.

## 5.2 Inštalácia a konfigurácia PAM modulov

Na koncových staniciach sa bude pre každú autentifikáciu používať samostatný PAM modul. Funkcia týchto modulov bola popísaná v kapitole 4.2. V tejto kapitole budú popísané kroky, ktoré je treba vykonať pre správnu funkciu jednotlivých PAM modulov.

### 5.2.1 PAM modul pre RADIUS

Oficiálny PAM modul, ktorý je možné nájsť na web stránkach FreeRADIUSu, nepodporuje autentifikáciu pre IPv6 adresu servera. Existuje však aj upravená neoficiálna verzia, ktorá má v sebe obsiahnutú podporu pre IPv6. Túto upravenú verziu vytvoril užívateľ, ktorý vystupuje pod prezývkou *KentaroAOKI* a je umiestnená na webových stránkach GitHUBu v repozitári pod názvom `pam-radius-ipv6`. GitHUB slúži pre vývojárov ako hosting pre zdrojové kódy. Pre verejné a open-source je bezplatný a umožňuje sa tak podieľať na vývoji softwaru komukoľvek na svete.

Prekvapujúco GitHUB nemá IPv6 adresu a keďže funguje na HTTPS protokole, nie je ani možné použiť SixXS bránu. Je teda potrebné stiahnuť archív s PAM modulom pre RADIUS na počítači, ktorý má aj IPv4 adresu. Aby bolo možné stiahnuť tento archív neskôr aj na klientských staniciach, ktoré majú pridelené iba IPv6 adresy, najvhodnejším riešením bude nahrať tento archív na náš server. Stiahnutý archív z GitHUBu má nič nehovoriaci názov `master.zip`. Preto je lepšie si ho nahrať na server s iným názvom, aby nám aj v budúcnosti bolo jasné, čo sa v archíve nachádza. Vykonáme to týmito príkazmi:

```
$ wget https://github.com/KentaroAOKI/pam-radius-ipv6/archive/master.
   zip
$ scp -6 master.zip student@[2001:718:1001:2c6::211]:~/radius-pam.zip
```

Keď budeme mať nahratý tento archív na serveri, môžme ho stiahnuť na klientských staniciach a rovno ho rozbaliť. Na kopírovanie opäť použijeme nástroj `scp`, a keďže sa jedná o *zip* archív, na jeho rozbalenie použijeme nástroj `unzip`. Po rozbalení môžme zmeniť adresár na ten, ktorý vznikol po rozbalení. Poslúžia nám na to tieto príkazy:

```
$ scp -6 student@[radius.raddp.cz]:~/radius-pam.zip ~/
$ unzip radius-pam.zip
$ cd pam-radius-ipv6-master/
```

Keďže archív obsahuje iba zdrojové kódy pre PAM modul, je potrebné ich skompilovať. Na to však ešte potrebujeme doinštalovať na klientských staniciach dva balíky, bez ktorých by kompilácia nebola možná. Prvý z nich je `g++`, čo je kompilátor pre jazyk C++. Druhý má názov `libpam0g-dev` a obsahuje vývojárske nástroje pre PAM moduly. Po inštalácii môžme spustiť príkaz `make`, ktorý vykoná kompiláciu. Tieto kroky vykonáme následujúcimi príkazmi: [17]

```
$ sudo apt-get install libpam0g-dev g++
$ make
```

Ak kompilácia prebehla v poriadku, objaví sa v aktuálnom adresári súbor s názvom `pam_radius_auth.so`, ktorý reprezentuje samotný PAM modul. Pre správnu funkčnosť je potrebné ho skopírovať do adresára obsahujúceho PAM moduly. Kde sa tento adresár nachádza, závisí od použitej architektúry. Ak máme v počítači 32-bitovú verziu Ubuntu, bude tento adresár umiestnený v `/lib/i386-linux-gnu/security/`. V prípade 64-bitového systému sa

jedná o adresár `/lib/x86_64-linux-gnu/security/`. V mojom prípade sa jedná o 64-bitovú verziu, takže budem súbor `pam_radius_auth.so` kopírovať do druhého uvedeného adresára. Ďalej je treba zmeniť práva tohoto súboru tak, aby k nemu mál prístup iba užívateľ `root`. Poslúžia nám na to tieto príkazy: [17]

```
$ sudo cp pam_radius_auth.so /lib/x86_64-linux-gnu/security/
$ sudo chmod 644 /lib/x86_64-linux-gnu/security/pam_radius_auth.so
```

Ďalším krokom je konfigurácia PAM modulu, aby vedel voči ktorému RADIUS serveru vykonať autentifikáciu. Pre tento modul je potrebné umiestniť konfiguráciu do adresára `/etc/raddb/` a nazvať ju `server`. V prípade, že adresár `/etc/raddb/` doposiaľ neexistuje, je potrebné ho vytvoriť. Do tohoto konfiguračného súboru môžme definovať aj viacero RADIUS serverov, a v prípade výpadku prvého by sa použil druhý. Ukážková konfigurácia sa nachádza pri zdrojových kódoch, a tak ju stačí skopírovať a premenovať na `server`. Nakoniec je potrebné súbor otvoriť a pridať do neho nami vytvorený RADIUS server. Vykonáme to touto sériou príkazov: [17]

```
$ sudo mkdir /etc/raddb
$ sudo cp pam_radius_auth.conf /etc/raddb/server
$ sudo nano /etc/raddb/server
...
[radius.raddp.cz]:1812 testing123 1
...
```

Taktiež sa odporúča zmeniť práva tohoto konfiguračného súboru tak, aby k nemu mal prístup iba užívateľ `root`. V inom prípade by mohol nejaký iný užívateľ upraviť tento súbor a nakonfigurovať doň podvrhnutý RADIUS server. Práva zmeníme pomocou príkazu `chmod` takto: [17]

```
$ sudo chmod 600 /etc/raddb/server
```

Po vykonaní vyššie uvedených krokoch môžme pristúpiť k testovaniu. Na otestovanie funkčnosti autentifikácie pomocou RADIUSu s využitím PAM modulu som si zvolil program `login`. Tento program sa využíva na prihlasovanie užívateľa do systému pomocou príkazového riadka. V adresári `/etc/pam.d/` sa nachádzajú konfiguračné súbory programov alebo služieb, ktoré využívajú PAM moduly. Konfiguračné súbory majú rovnaký názov ako služba, ktorá ich využíva. My teda potrebujeme upraviť súbor `login`. Otvoríme ho a aby sa PAM modul pre RADIUS použil ako prvý v poradí pre autentifikáciu, vložíme potrebnú konfiguráciu pred riadok obsahujúci `@include common-auth`. Otvoríme teda súbor `login` a vložíme do neho potrebnú konfiguráciu takto: [17]

```
$ sudo nano /etc/pam.d/login
...
auth  sufficient  pam_radius_auth.so debug
```

```
@include common-auth
...
```

Kľúčové slovo `debug` znamená, že sa bude zapisovať priebeh autentifikácie do súboru `/var/log/auth.log`. Pre správne fungovanie PAM modulu sa v konfigurácii slovo `debug` nachádzať nemusí, je to však veľmi užitočné pri odstraňovaní problémov s nefunkčnou autentifikáciou. Užitočné je aj zisťovanie prípadných problémov na RADIUS serveri. Sú dve základné možnosti ako sledovať udalosti na serveri. Prvá z nich je spustiť FreeRADIUS v debugovacom móde pomocou príkazu `freeradius -X`. V tomto móde bude FreeRADIS vypisovať všetky udalosti, ktoré sa na ňom dejú. Často je to však viac informácii ako potrebujeme. Druhou možnosťou je zapnúť logovanie prihlasovaní a vtedy bude FreeRADIUS zapisovať pokusy o prihlásenie do súboru `radius.log`, ktorý sa nachádza v adresári `/var/log/freeradius/`. V pôvodnom nastavení je táto možnosť vypnutá a je potrebné ju zapnúť v konfiguračnom súbore `radiusd.conf`, ktorý sa nachádza v zložke s konfiguračnými súbormi FreeRADIUSu. Otvoríme teda tento súbor a na riadku 343 zmeníme hodnotu parametru `auth = no` na `auth = yes`. Po tejto zmene je ešte treba reštartovať službu `freeradius`. Potom keď budeme chcieť na serveri sledovať pokusy o prihlásenie, môžme na to využiť utilitu `tail`, ktorá slúži na zobrazenie posledných riadkov súboru. Spustíme ju s parametrom `-f`, ktorým nastavíme, že sa má výstup tohto programu automaticky aktualizovať vždy, keď sa do súboru niečo zapíše. Vykonáme to týmito príkazmi a úpravou konfiguračného súboru:

```
$ sudo nano /etc/freeradius/radiusd.conf
...
      auth = yes
...
$ sudo service freeradius restart
$ sudo tail -f /var/log/freeradius/radius.log
```

Ďalej je potrebné aby sa užívateľ, ktorého chceme overiť na RADIUS servery nachádzal aj lokálne na klientskej stanici. Stačí pridať užívateľa bez hesla a vytvoriť mu domovskú zložku. Tu nám môže opäť poslúžiť užívateľ *bob* a nástroj `useradd`, ktorý slúži na pridávanie užívateľov. Spustíme ho s parametrom `-m`, ktorým definujeme, aby sa vytvoril aj domovský adresár pre tohoto užívateľa. Ak chceme aby užívateľ mohol spúšťať programy s oprávnením užívateľa `root`, pridáme ho pomocou parametra `-G` do skupiny `sudo`. Ak bude daný užívateľ využívať príkazovú riadku, je vhodné mu definovať tzv. „shell“, ktorý slúži na spracovanie zadaných príkazov a ich odovzdanie operačnému systému na vykonanie. Najpoužívanejší „shell“ v linuxe je `bash`. Keby sme nedefinovali pri vytváraní užívateľa aby sa použil `bash`, bol by použitý „shell“ s názvom `sh`. Je to predchodca `bash` a chýba v ňom mnoho funkcií, ako napríklad história použitých príkazov alebo doplňovanie príkazov pomocou tabulátoru. Na pridanie užívateľa *bob* s vypnutým heslom a vyššie uvedenými parametrami použijeme tento príkaz: [18]

```
$ sudo useradd -m -G sudo -s /bin/bash bob
```

Keď už máme užívateľa vytvoreného, môžme pristúpiť k otestovaniu funkčnosti autentifikácie na RADIUS serveri. Na prihlásenie použijeme program `login`. Príkaz bude vyzerať takto:

```
$ sudo login bob
```

Po spustení tohto príkazu budeme vyzvaný na zadanie hesla pre užívateľa bob. Ako heslo použijeme „hello“, ktoré je nastavené v konfiguračnom súbore `users.conf` na FreeRADIUSe. Ak boli všetky nastavenia správne, malo by úspešne autentifikovať užívateľa a prihlásiť ho do systému. Takéto úspešné prihlásenie je zobrazené na obrázku 18. Zároveň by sa na RADIUS serveri mal v logovacom súbore zobraziť následujúci riadok, ktorý potvrdzuje úspešnú autentifikáciu:

```
Fri Apr 8 21:02:27 2016 : Auth: (0) Login OK: [bob] (from client
   private_ipv6_subnet port 11211)
```

```
student@client:~$ sudo login bob
Password:
Last login: Thu Apr 21 14:04:00 PDT 2016 on pts/0
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 4.2.0-27-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

bob@client:~$
```

Obr. 18: Úspešné prihlásenie pomocou PAM modulu pre RADIUS.

FreeRADIUS umožňuje na autentifikáciu využiť aj užívateľov, ktorý sú definovaný ako klasický systémový užívatelia v Linuxe. Vykonáva kontrolu hesiel pomocou PAM frameworku. V pôvodnom nastavení je táto možnosť vypnutá a je ju preto potrebné povoliť. Na povolenie použijeme dostupný modul FreeRADIUSu s názvom `pam`, ktorý sa nachádzajú v adresári `/etc/freeradius/mods-available/`. Ako prvý krok je vytvorenie symbolického odkazu, ktorým nastavíme, aby FreeRADIUS server tento modul používal. Slúži na to nástroj `ln` s prepínačom `-s`, ktorým definujeme, že sa jedná práve o symbolický odkaz. Odkaz vytvoríme v adresári, v ktorom sa nachádzajú aktívne moduly. Na to použijeme tento príkaz: [19, 20]

```
$ sudo ln -s /etc/freeradius/mods-available/pam /etc/freeradius/mods-
   enabled/pam
```

To však nestačí a je potrebné ešte povoliť autentifikáciu týmto modulom v konfiguračnom súbore `default`, ktorý sa nachádza v adresári `/etc/freeradius/sites-enabled/`. Otvoríme teda tento súbor a nájdem v ňom časť `authenticate`, v ktorej odkomentujeme riadok `pam`. V mojej verzii FreeRADIUSu sa jednalo o riadok 484. Predposledným krokom je definovať buď, že má FreeRADIUS pri konkrétnom užívateľovi použiť PAM modul alebo, že sa pre

všetkých užívateľoch použiť tento typ autentifikácie. Ak sa má použiť pre všetkých pridáme do konfiguračného súboru `users` riadok `DEFAULT Auth-Type := PAM`. Ak sa má použiť iba pre konkrétneho užívateľa, napr. v mojom prípade užívateľa `student`, pridáme tam riadok `student Auth-Type := PAM`. Nakoniec ešte reštartujeme FreeRADIUS pre aplikovanie daných zmien. Všetko vykonáme následujúcimi príkazmi a úpravou konfiguračných súborov: [19]

```
$ sudo nano /etc/freeradius/sites-enabled/default
...
authenticate {

      # Pluggable Authentication Modules.
      pam
...
}
$sudo nano /etc/freeradius/users
...
student Auth-Type := PAM
...
$ sudo service freeradius restart
```

Ak chceme otestovať funkčnosť tejto konfigurácie, vystačíme si s nástrojom `radtest`. Použijeme teda známy príkaz, iba so zmeneným prihlasovacím menom a heslom. V mojom prípade je meno aj heslo rovnaké a to *student*. Príkaz teda bude vyzerať takto:

```
$ radtest -6 student student ip6-localhost 0 testing123
```

Ak bola autentifikácia úspešná, server nám odpovie správou Access-Accept a v logovacom súbore sa objaví tento riadok:

```
Fri Apr 8 22:33:26 2016 : Auth: (0) Login OK: [student] (from client
   localhost_ipv6 port 0)
```

### 5.2.2 PAM modul pre Google Authenticator

Pomocou tohoto PAM modulu, môžme na autentifikáciu užívateľa využiť jednorázové heslo vygenerované pomocou mobilnej aplikácie Google Authenticator. Jednorázové heslo je generované s využitím otvoreného štandardu, ktorý bol vyvinutý iniciatívou pre otvorenú autentifikáciu OATH. Umožňuje generovať OTP, ktoré sú časovo závislé (TOTP) alebo založené na počítadle použití (HOTP). Pri použití TOTP sa generuje nové heslo v pravidelnom časovom intervale, kdežto pri HOTP sa generuje nové heslo na vyžiadanie. Platnosť HOTP hesla teda vyprší až pri jeho použití. Existujú dve možnosti ako implementovať tento PAM modul, keď

ho chceme využívať v kombinácii s FreeRADIUSom a budú popísané v nasledujúcich podkapitolách. [21]

#### 5.2.2.1 Inštalácia Google Authenticatora a PAM modulu

Inštalácia je pri obidvoch spôsoboch implementácie rovnaká, a preto ju popíšem samostatne. Google Authenticator je vyvíjaný ako open-source na GitHube. Vďaka tomu je jeho vývoj veľmi rýchly a v mojom prípade bola posledná verzia stará iba 22 hodín. Ako už bolo v predchádzajúcej kapitole spomenuté, webové stránky GitHubu fungujú iba na IPv4. Preto je opäť nutné stiahnuť zdrojové kódy na PC, ktoré má aj IPv4 adresu a nakopírovať stiahnutý súbor na náš server, aby sme ho potom prípadne mohli sťahovať aj na klientských staniciach. Použijeme na to podobný príkaz ako v predchádzajúcej kapitole a bude vyzerať takto:

```
$ wget https://github.com/google/google-authenticator/archive/master.
   zip
$ scp -6 master.zip student@[2001:718:1001:2c6::211]:~/google-auth.
   zip
```

Ak budeme inštalovať Google Authenticator priamo na serveri, môžme rovno rozbaliť archív pomocou nástroja `unzip` a zmeniť adresár na novo vzniknutý po rozbalení archívu. Konkrétne sa jedná o podadresár libpam, v ktorom sa nachádzajú zdrojové kódy pre Google Authenticator a jeho PAM modul. Ak budeme Google Authenticator inštalovať na klientských staniciach, najprv ešte archív stiahneme zo servera pomocou utility `scp`. Séria príkazov sa, okrem názvov súborov, od tej z predchádzajúce kapitoly nelíši a bude vyzerať takto:

```
$ scp -6 student@[radius.raddp.cz]:~/google-auth.zip ~/
$ unzip google-auth.zip
$ cd google-authenticator-master/libpam/
```

Pred samotnou kompiláciou a inštaláciou je ešte potrebné doinštalovať z repozitárov Ubuntu dva balíky. Prvý z nich je `dh-autoreconf` a je to prídavný modul pre nástroj `debhelper`. Druhý, ktorý má názov `qrencode` nie je nevyhnutný na kompiláciu, ale umožňuje nám pri vytváraní hesla slúžiaceho na generovanie OTP zobraziť QR kód v príkazovom riadku. Tento QR kód môžme potom naskenovať do mobilnej aplikácie Google Authenticator namiesto ručného zadávania pomerne dlhého hesla. Po nainštalovaní týchto závislostí, môžme pristúpiť ku kompilácii PAM modulu a inštalácii nástroja `google-authenticator`. Vykonáme to spustením týchto príkazov: [22]

```
$ sudo apt-get install dh-autoreconf qrencode
$ ./bootstrap.sh
$ ./configure
$ make
$ sudo make install
```

Po kompilácii sa vytvorí adresár `.libs`, v ktorom sa nachádza samotný PAM modul. Ten musíme skopírovať do adresára, v ktorom sa nachádzajú všetky PAM moduly. Kde sa tieto adresáre nachádzali som už spomínal v predchádzajúcej kapitole. Taktiež je potrebné zmeniť oprávnenia pre tento súbor, aby mal k nemu prístup iba užívateľ `root`. Použijeme na to tieto dva príkazy:

```
$ sudo cp .libs/pam_google_authenticator.so /lib/x86_64-linux-gnu/
   security/
$ sudo chmod 644 /lib/x86_64-linux-gnu/security/
   pam_google_authenticator.so
```

#### 5.2.2.2 Inštalácia NTP servera

Tým máme základnú inštaláciu Google Authenticatora dokončenú. Keďže však budeme používať časovo závislé jednorázové heslá, je potrebné zaistiť na serveri aj klientských staniciach presný čas. Pre NTP server využijeme balík `ntp` z repozitárov Ubuntu. Zároveň nám poslúži aj ako klient na koncových staniciach. Nainštalujeme teda tento balík na serveri aj klientskej stanici pomocou príkazu:

```
$ sudo apt-get install ntp
```

Keďže naša sieť je postavená iba na IPv6, je potrebné nakonfigurovať takzvaný NTP pool, ktorý podporuje IPv6 adresy. Na serveri teda otvoríme konfiguračný súbor `/etc/ntp.conf` a zakomentujeme alebo odstránime NTP servery, ktoré sa tu nachádzajú v pôvodnom nastavení. Napokon do konfiguračného súboru vložíme NTP pool `ntp.eu.sixxs.net`, na ktorom sú dostupné NTP servery aj na IPv6 adresách. Za týmto serverom doplníme parameter `iburst`, ktorým zaistíme aby sa hneď pri reštarte NTP služby synchronizoval čas s nami definovaným poolom. Napokon službu reštartujeme. Vykonáme to týmito príkazmi a úpravou súboru: [23, 24]

```
$ sudo nano /etc/ntp.conf
...
#server 3.ubuntu.pool.ntp.org
server ntp.eu.sixxs.net iburst
...
$ sudo service ntp restart
```

Na klientskej strane spravíme to isté, s výnimkou nastaveného NTP servera v konfiguračnom súbore. Tam namiesto servera `ntp.eu.sixxs.net` nastavíme doménové meno nášho servera, a to je `radius.raddp.cz`. Po reštarte si môžme našu konfiguráciu overiť pomocou príkazu: [24]

```
$ sudo ntpq -p
```

Vo výstupe by sa nám mala objaviť IP adresa alebo doménové meno NTP servera, pomocou ktorého prebehla synchronizácia. Výstup tohto testovacieho príkazu zo strany servera a klienta, je možné vidieť na obrázku 19. V prípade mobilného zariadenia využíva aplikácia Google Autheticator vnútorné hodiny, ktoré sú automaticky synchronizované s časom na serveroch Googlu.

```
student@radius:~$ sudo ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*gblon02.sixxs.n 205.1.156.195    3 u   16   64    1   36.563  116.249   3.080
student@radius:~$
```

```
student@client:~$ sudo ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*ns.raddp.cz     77.75.104.126    4 u   34   64    1    0.249   75.157   1.103
student@client:~$
```

Obr. 19: Overenie funkčnosti nastavenia NTP. Vyššie na serveri a nižšie na klientovi.

#### 5.2.2.3 Google Authenticator lokálne

Prvou možnosťou ako implementovať Google Authenticator je lokálne na užívateľskej stanici. To znamená, že pri dvojfaktorovej autentifikácii s využitím FreeRADIUS servera, by sa užívateľ najprv autentifikoval voči RADIUS serveri, a potom by prebehla lokálna autentifikácia s využitím PAM modulu pre Google Authenticator.

Všetkým užívateľom na koncovej stanici je potrebné vytvoriť heslo, ktoré sa bude používať na generovanie jednorázového hesla. Vykonáva sa to spustením príkazu `google-authenticator` pod užívateľom, pre ktorého chcem heslo vytvoriť. Po spustení tohoto príkazu sa nás sprievodca spýta aký typ OTP sa má generovať pre daného užívateľa, potom či chceme aktualizovať súbor `.google_authenticator`, ktorý je umiestnený do domovského adresára užívateľa. Do tohoto súboru sa ukladá vygenerované heslo spolu so záložnými kódmi, ktoré je možné použiť v prípade, že nemáme k dispozícii mobilnú aplikáciu s aktuálnym OTP. Je to veľmi užitočné v prípade straty mobilného telefónu, v ktorom sme mali nakonfigurovanú aplikáciu Google Authenticator. Na prihlásenie potom užívateľ použije záložný kód a môže si znova vygenerovať nové heslo pre OTP.

Ďalej sa nás pýta, či chceme obmedziť viacnásobné použitie jedného OTP. To znamená, že každé jednorázové heslo môžme použiť iba raz, čo síce obmedzí možnosť prihlásiť sa do systému iba raz za 30 sekúnd, ale je to veľmi účinná ochrana proti takzvanému „man-in-the-middle“ útoku. Predposlednou otázkou je, či chceme rozšíriť tolerančné okno z defaultných 90 sekúnd na 4 minúty. Toto tolerančné okno slúži ako kompenzácia pre možný rozdiel v čase na koncovej stanici a v mobilnom zariadení. Preto je potrebné pri metóde časovo závislého OTP zaistiť, aby mali obidve zariadenia nastavený rovnaký čas. Poslednou otázkou je či chceme

obmedziť počet pokusov o prihlásení na maximálne 3 každých 30 sekúnd. Slúži to ako ochrana proti tzv. „brute-force“ útoku.

Pre väčšie pohodlie, hlavne pri konfigurácii viacerých užívateľoch naraz, môžme spustiť príkaz `.google_authenticator` s parametrami, ktoré nastavia nami nadefinované požiadavky bez nutnosti ich konfirmácie. Tie parametre sú nasledujúce:

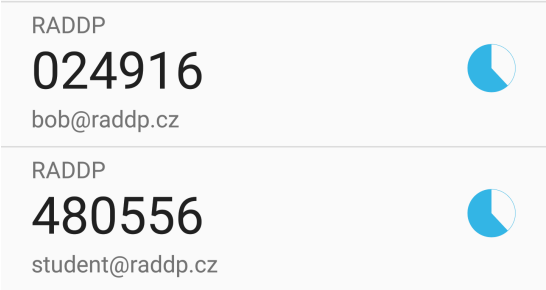
- **`--qr-mode={NONE,ANSI,UTF8}`** - nastavíme ním štandard, ktorý sa použije pri generovaní QR kódu v príkazovom riadku. V prípade nastavenia tohoto paramateru na `NONE`, sa nebude QR kód generovať. Mne sa osvedčilo použiť štandard UTF8, pri ktorom sa zobrazí celý QR kód v príkazovom riadku narozdiel od štandardu ANSI, ktorý vygeneroval príliš veľký nečitateľný kód.
- **`-t`** - týmto parametrom zvolíme, že sa má použiť časovo závislé jednorázové heslo TOTP.
- **`-d`** - tento parameter zakazuje opätovné použitie toho istého TOTP.
- **`-f`** - nastavíme ním, aby sa zapísal konfiguračný súbor do domovského adresára užívateľa bez nutnosti potvrdenia.
- **`--window-size=W`** - tento parameter určuje veľkosť okna platných kódov v minútach.
- **`--rate-limit=N`** - týmto parametrom nastavíme maximálny počet prihlásení N v sekundách.
- **`--rate-time=M`** - určíme ním za aký čas M môžme znova opakovať počet prihlásení N a udáva sa v sekundách.
- **`--issuer=<issuer>`** - určíme ním, pod akým názvom sa bude zobrazovať daný záznam v mobilnej aplikácii Google Authenticator po oskenovaní QR kódu. Je veľmi užitočné použiť tento prepínač, aby sme vedeli rozlíšiť, ktorý záznam patrí ktorej službe.
- **`--label=<label>`** - tiež slúži na odlíšenie záznamov. V tomto prípade je užitočný ak máme v jednej službe viacero účtov.

V mojom prípade bude teda pre užívateľa *bob* použitý tento príkaz:

```
$ google-authenticator --qr-mode=UTF8 -t -d -f --window-size=4 --rate
   -limit=3 --rate-time=30 --issuer=RADDP --label=bob@raddp.cz
```

Výstupom tohoto programu bude vygenerovaný QR kód zobrazený priamo v príkazovom riadku a URL adresa, k tomu istému QR kódu. Ďalej sa tu nachádza heslo použité na generovanie OTP, ktoré môžeme prípadne zadať ručne do mobilnej aplikácie, namiesto skenovanie QR kódu. Je to však metóda časovo náročnejšia, menej pohodlnejšia, ale predovšetkým náchylná na chybu keďže toto heslo je pomerne dlhé a zložené s písmen a číslic. Vo výstupe sú vypísané aj záložné kódy. Výstup tohoto príkazu zobrazuje obrázok 40 v prílohe C.

Vygenerovaný QR kód môžme oskenovať v mobilnej aplikácii Google Authenticator a po úspešnom oskenovaní, sa nám ihneď zobrazí v zozname OTP hesiel. Ukážka rozhrania aplikácie Google Authenticator je zobrazená na obrázku 20. Tu je možné vidieť názov služby a účtu, tak ako sme ich definovali pomocou vyššie uvedených parametrov.




Obr. 20: Ukážka OTP hesiel programe Google Authenticator.

Na otestovanie funkčnosti znova využijeme program `login` a užívateľa *bob*. Opäť budeme upravovať konfiguračný súbor `/etc/pam.d/login`. Z predchádzajúcej konfigurácie v ňom už máme definované použitie PAM modulu pre RADIUS na autentifikáciu. Zmeníme pri ňom kľúčové slovo `sufficient` na `requisite`, ktorým zaistíme, že pri neúspešnej autentifikácii na RADIUS serveri, nebude možné pokračovať v d'alšej autentifikácii a tá automaticky zlyhá. Ako d'alší riadok pridáme konfiguráciu PAM modulu pre Google Authetication, ktorému definujeme kľúčové slovo sufficient a na koniec mu pridáme parameter `debug`. Týmto parametrom zaistíme, podobne ako u RADIUS PAM modulu, aby sa do logovacieho súboru `auth.log` zapisovali udalosti pri volaní tohoto PAM modulu. Ďalším parametrom `echo_verification_code` definujeme, aby sa pri písaní jednorázového hesla toto heslo zobrazovalo. Keďže máme zapnutú ochranu, ktorá nám umožňuje použiť každý kód iba raz, na bezpečnosť to nebude mať žiaden vplyv. Zobrazovaním hesla sa navyše vyhneme neúspešným autentifikáciám, ktoré môžu vzniknúť chybným zadaním OTP. Posledným parametrom je `nullok`, ktorý nám umožní prihlásenie užívateľa, ktorý ešte nemá nakonfigurovaný Google Authenticator. Vykonáme to týmto príkazom a úpravou spomenutého súboru: [22]

```
$ sudo nano /etc/pam.d/login
...
auth  requisite  pam_radius_auth.so debug
auth  sufficient pam_google_authenticator.so debug
   echo_verification_code nullok
@include common-auth
...
```

Napokon pomocou príkazu `$sudo login bob` vyvoláme prihlasovanie užívateľa *bob* a budeme vyzvaný najprv na zadanie hesla, ktoré sa overí voči RADIUS serveru, a potom na

zadanie OTP z mobilnej aplikácie Google Authenticator. Takéto prihlásenie je zobrazené na obrázku 21.

```
student@client:~$ sudo login bob
Password:
Verification code: 531288
Last login: Thu Apr 21 14:28:54 PDT 2016 on pts/0
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 4.2.0-27-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

bob@client:~$
```

Obr. 21: Overenie funkčnosti dvojfaktorovej autentifikácie s použitím Google Authenticatora.

#### 5.2.2.4 Google Authenticator na FreeRADIUS serveri

Ďalšou možnosťou ako použiť Google Authenticator je jeho implementácia priamo na FreeRADIUS serveri. Má to niekoľko výhod, ale aj nevýhod. Medzi výhody môžme zaradiť možnosť použiť dvojfaktorovú autentifikáciu aj so zariadeniami, ktoré priamo nepodporujú autentifikáciu pomocou PAM modulov. Ďalšou výhodou je, že pri tejto implementácii stačí Google Authenticator nainštalovať iba na FreeRADIUS serveri. Má to však aj svoje nevýhody. Hlavnou z nich je, že keďže PAM modul pre Google Authenticator pracuje priamo s linuxovými užívateľmi, musia byť všetci užívatelia, ktorých je potrebné takto autentifikovať, definovaný ako systémový užívatelia. Nie je teda možné používať na uchovávanie užívateľov žiadnu inú databázu. Medzi nevýhody môžme zaradiť aj to, že pri autentifikácii užívateľa je potrebné za klasické užívateľské heslo doplniť vygenerované jednorázové heslo, čo nie je príliš užívateľsky prívetivé.

Po nainštalovaní nástroja `google-authenticator`, je ešte potrebné upraviť konfiguráciu FreeRADIUSU. Najprv povolíme autentifikáciu pomocou PAM modulu, tak ako bola popísaná v kapitole o PAM module pre RADIUS. FreeRADIUS si pri inštalácii vytvorí užívateľa a skupinu s menom `freerad`, a je spúšťaný práve pod týmto užívateľom. Na to aby však FreeRADIUS mohol pristupovať k súboru `.google_authenticator`, ktorý sa nachádza v domovskom adresári každého užívateľa, musí sa spustiť po užívateľom `root`. To nastavíme v konfiguračnom súbore `radiusd.conf`. Tento súbor otvoríme a v oddieli `security` nastavíme aby sa spúšťal pod užívateľom a skupinou s názvom `root`: [25]

```
$ sudo nano /etc/freeradius/radiusd.conf
security {
...
      user = root
      group = root
...
}
```

Ako ďalšie je treba v súbore `users` definovať, že sa má pri autentifikácii užívateľa vždy použiť PAM modul. Definujeme to pridaním následujúceho riadka do tohoto súboru: [25]

```
$ sudo nano /etc/freeradius/users
...
DEFAULT     Auth-Type := PAM
...
```

Následne je potrebné upraviť konfiguračný súbor PAM modulu `/etc/pam.d/radiusd`. Zakomentujeme v ňom všetky riadky z pôvodnej konfigurácie a definujeme v ňom, ktoré PAM moduly sa majú použiť. Ako prvý sa bude používať PAM modul pre Google Authenticator, pri ktorom je dôležitý parameter `forward_pass`, ktorým definujeme, že po autentifikácii týmto modulom sa má systémové heslo bez OTP hesla odovzdať nasledujúcemu PAM modulu. Ten bude v našom prípade `pam_unix.so` a je nutné aby bol pri ňom parameter `use_first_pass`, ktorým nastavíme, že nemá o heslo žiadať užívateľa, ale má sa použiť heslo z predchádzajúceho PAM modulu. Súbor `radiusd` teda upravíme následovne: [22, 25]

```
$ sudo nano /etc/pam.d/radiusd
...
#@include common-auth
#@include common-account
#@include common-password
#@include common-session

auth requisite pam_google_authenticator.so debug forward_pass nullok
auth required pam_unix.so use_first_pass
```

Na otestovanie funkčnosti využijem užívateľa *student*, pre ktorého vygenerujem súbor `.google_authenticator`, v ktorom je uložené heslo na generovanie jednorázových hesiel. Použijem na to príkaz `google-authenticator` s parametrami aké boli spomenuté už vyššie. Jediný rozdiel od predchádzajúceho príkazu je v parametri `label`, v ktorom definujeme užívateľa *student*, aby sme mohli rozlíšiť záznamy v mobilnej aplikácii Google Authenticator. Ako poslednú vec je ešte potreba reštartovať samotnú službu `freeradius`. Použijeme teda tieto príkazy: [25]

```
$ google-authenticator --qr-mode=UTF8 -t -d -f --window-size=4 --rate
   -limit=3 --rate-time=30 --issuer=RADDP --label=student@raddp.cz
$ sudo service freeradius restart
```

Pre overenie funkčnosti tejto konfigurácie si vystačíme s utilitou `radtest` na klientskej stanici. Syntax tohoto príkazu tu už bola niekoľkokrát spomenutá. Za zmienku stojí iba to, že systémové heslo zadávame dokopy spolu s heslom vygenerovaným pomocou mobilnej aplikácie. Po overení jednorázového hesla odovzdá PAM modul Google Authenticatora následujú-

cemu modulu už iba systémové heslo. Overenie pomocou utility `radtest` z klientskej stanice je zobrazené na obrázku 22.

```
student@client:~$ radtest -6 student student334515 radius.raddp.cz 0 testing123
Sent Access-Request Id 81 from [::]:33989 to [2001:718:1001:2c6::211]:1812 leng
h 89
        User-Name = "student"
        User-Password = "student334515"
        NAS-IPv6-Address = 2001:718:1001:2c8:76d4:35ff:fe7c:ece
        NAS-Port = 0
        Message-Authenticator = 0x00
        Cleartext-Password = "student334515"
Received Access-Accept Id 81 from [2001:718:1001:2c6::211]:1812 to [::]:0 lengt
 20
student@client:~$
```

Obr. 22: Overenie funkčnosti Google Authenticatora na FreeRADIUS serveri.

### 5.2.3 PAM modul pre čítačku odtlačkov prsta

Biometrickú metódu autentifikácie bude v mojom navrhnutom riešení reprezentovať autentifikácia pomocou odtlačku prsta. Na to využijem čítačku odtlačkov prsta *Upek Eikon II*, ktorá je vyobrazená na obrázku 23.


Obr. 23: Čítačka odtlačkov prsta Upek Eikon II.

Po pripojení čítačky do počítača si môžme overiť jej úspešné rozpoznanie operačným systémom. Na Ubuntu nám na to poslúži nástroj `lsusb`, ktorý slúži na vypísanie zariadení pripojených pomocou rozhrania USB. Tento výpis môže byť v niektorých prípadoch dlhý, a tým pádom menej prehľadný. Preto môžme presmerovať výstup tohoto nástoja do programu `grep`, pomocou ktorého vypíšeme iba zariadenia obsahujúce v názve názov firmy *Upek*. Príkaz, ktorý na to použijeme a jeho výstup v prípade úspešného rozpoznania bude následovný:

```
$ lsusb | grep Upek
```

```
Bus 002 Device 004: ID 147e:2016 Upek Biometric Touchchip/Touchstrip
   Fingerprint Sensor
```

Pre prácu s čítačkou využijeme open-source program s názvom fprint, pre ktorý existuje aj PAM modul. Na inštaláciu tohoto PAM modulu využijeme repozitáre Ubuntu. Pri inštalácii balíka s PAM modulom s názvom `libpam-fprintd`, sa spolu s ním nainštalujú aj jeho závislostí, medzi ktorými je aj balík `fprintd`. Jedná sa o takzvaný „daemon“, ktorý poskytuje možnosť skenovania odtlačkov prstov pomocou zbernice D-Bus. Nainštalujeme teda PAM modul pomocou príkazu: [26, 27]

```
$ sudo apt-get install libpam-fprintd
```

Po dokončení inštalácie môžme začať so skenovaním odtlačkov. Vykonáva sa to spustením príkazu `fprintd-enroll` pod užívateľom, pre ktorého chceme naskenovať odtlačok prsta alebo ak chceme skenovať odtlačok pre iného užívateľa, spustíme príkaz v tvare `$ sudo fprintd-enroll [username]`. Keď spustíme tento príkaz bez parametrov, vyzve nás na zosnímanie pravého ukazováka. Príkaz však môžme spustiť s parametrom `-f` doplnený názvom prsta, ktorým určíme presne aký prst chceme naskenovať. Všetky možnosti, ktoré môžme použiť v tomto parametri, sú vypísané nižšie v poradí od ľavého malíčka až po pravý malíček: [27]

```
left-little-finger, left-ring-finger, left-middle-finger, left-index-
   finger, left-thumb, right-thumb, right-index-finger, right-middle-
   finger, right-ring-finger, right-little-finger
```

Naskenovať môžme aj viac prstov, je však treba mať na pamäti, že PAM modul použije vždy prvý oskenovaný prst. Preto pri zmene prsta, ktorý chceme použiť pri autentifikácii, je treba najprv vymazať všetky staršie záznamy a až potom oskenovať nový prst. Staré záznamy sa vymažú pomocou tohto príkazu, kde za *username* doplníme meno užívateľa pre ktorého chceme záznam vymazať:

```
$ fprintd-delete [username]
```

Keďže väčšina užívateľov má na pravej strane počítačovú myš, budem na skenovanie používať ľavý ukazovák. Vykonáme to pomocou príkazu

```
$ fprintd-enroll -f left-index-finger
```

Po spustení príkazu musíme 5 krát prejsť ľavým ukazovákom cez snímač na čítačke. Niekedy sa stane, že nemáme prst v strede čítačky, prípadne prejdeme cez čítačku len krátkou plochou prsta. Ak tento problém nastane, program nás na to upozorní konkrétnou hláškou a musíme toto skenovanie zopakovať. Proces skenovania aj s ukážkou chybových hlášok zobrazuje obrázok 24.

Na otestovanie funkčnosti nám ako aj v predchádzajúcich prípadoch poslúži program `login`. Upravíme teda konfiguráciu pre jeho PAM modul a pridáme do nej riadok na použitie PAM

```
bob@client:~$ fprintd-enroll -f left-index-finger
Using device /net/reactivated/Fprint/Device/0
Enrolling left-index-finger finger.
Enroll result: enroll-stage-passed
Enroll result: enroll-stage-passed
Enroll result: enroll-swipe-too-short
Enroll result: enroll-finger-not-centered
Enroll result: enroll-stage-passed
Enroll result: enroll-stage-passed
Enroll result: enroll-completed
bob@client:~$
```

Obr. 24: Ukážka skenovania odtlačku prsta.

modulu pre čítačku odtlačkov prsta. Pridáme ho za pridanú konfiguráciu z predchádzajúcich kapitol a budeme mať teda trojfaktorovú autentifikáciu. Úspešný test takejto konfigurácie, je možné vidieť na obrázku 25. Vykonáme to týmto príkazom a úpravou konfiguračného súboru `login`:

```
$ sudo nano /etc/pam.d/login
...
auth  requisite  pam_radius_auth.so debug
auth  requisite  pam_google_authenticator.so debug
   echo_verification_code nullok
auth  sufficient pam_fprintd.so debug
@include common-auth
...
```

```
student@client:~$ sudo login bob
Password:
Verification code: 331369
** Message: debug on
Swipe your left index finger across the fingerprint reader
Last login: Thu Apr 21 14:45:33 PDT 2016 on pts/0
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 4.2.0-27-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

bob@client:~$
```

Obr. 25: Ukážka úspešnej troj-faktorovej autentifikácie s využitím čítačky odtlačku prstov.

Kľúčovým slovom `sufficient` zabezpečíme, aby sa po úspešnej autentifikácii týmto PAM modulom skončila úspešne aj celá autentifikácia. V prípade neúspešnej autentifikácie sa však pokračuje na ďalší PAM modul, ktorý je v tomto prípade zahrnutý v `common-auth`, a je to overenie systémového hesla. V našom prípade užívateľ *bob*, nemá pridelené žiadne systémové heslo, čo by v prípade, že by si poranil prst, a nebola by teda možná autentifikácia jeho odtlačkom, znamenalo, že sa do systému neprihlási. Preto je vhodné na každej klientskej stanici vy-

tvoriť nejakého užívateľa, ktorý bude slúžiť iba správcovi a bude mať pridelené okrem hesla na RADIUSe, aj systémové heslo. Tým zabezpečíme, aby sa aj v prípade nemožnosti autentifikácie odtlačkom prsta tohto správcu, mohol prihlásiť do systému a oskenovať užívateľovi, prípadne aj sebe, iný prst na autentifikáciu.

## 5.3 Inštalácia a konfigurácia MySQL databázy

FreeRADIUS sa môže pripájať na SQL databázu pre získanie údajov o užívateľoch. FreeRADIUS má v sebe implementovanú podporu pre množstvo rôznych programov na správu SQL databázy. Jednou z nich je aj MySQL databáza, ktorá je veľmi populárna medzi užívateľmi a je šírená pod open-source licenciou. MySQL je jednoduchá na konfiguráciu a vzhľadom na veľkú komunitu, ktorá túto databázu používa, aj veľmi dobrú podporu. Do tejto databázy sa najčastejšie ukladajú užívateľské prihlasovacie údaje. Taktiež tu môžme definovať NAS klientov, no v mojom prípade budem túto databázu využívať iba pre ukladanie užívateľských účtov. [1]

Ukladanie užívateľských údajov do SQL databáza so sebou prináša viacero výhod v porovnaní s ich ukladaním do konfiguračného súboru `users`. Najdôležitejšie z nich sú: [1]

- **Škálovateľnosť** - databáza sa môže nachádzať na inom serveri, vďaka čomu nemusí byť uložená priamo na FreeRADIUS serveri. Túto vlastnosť je dobré využiť aj keď máme záložný FreeRADIUS server. V takom prípade stačí užívateľoch uložiť do jednej databázy, ktorú budú využívať obidva FreeRADIUS servery.
- **Užívateľská prívetivosť** - existuje mnoho webovo založených programov, pomocou ktorých môžeme spravovať MySQL databázu. Medzi najznámejšie patrí napríklad phpMyAdmin.
- **Flexibilita** - užívatelia môžu byť pridávaný do databázy za behu, bez nutnosti zakaždým reštartovať FreeRADIUS.
- **Manažovateľnosť** - užívatelia môžu byť pridávaný do skupín, pomocou ktorých môžme spravovať ich spoločné atribúty.

Na inštaláciu MySQL databázy využijem repozitáre Ubuntu. Ak sme si pri inštalácii FreeRADIUS servera zvolili iba nutné balíky na funkčnosť tohto servera, je ešte potrebné doinštalovať balík, ktorý obsahuje modul pre MySQL. Tento balík má názov `freeradius-mysql` a nainštalujeme ho z nami vytvorených inštalačných balíkov. Pri inštalácii MySQL servera z repozitárov budeme vyzvaný na zadanie hesla pre správu MySQL databázy. Je preto treba nastaviť toto heslo dostatočne bezpečné. Inštaláciu uvedených balíkov vykonáme príkazom: [1]

```
$ sudo dpkg -i freeradius-mysql_3.0.11+git_amd64.deb
$ sudo apt-get install mysql-server
```

Po dokončení inštalácie sa prihlásime na MySQL server pod užívateľom `root` a vytvoríme databázu s názvom `radius`. Použijeme na to príkaz `mysql` a prepínačom `-u` určíme, že sa má prihlásiť pod týmto užívateľom. Ďalší parameter použijeme `-p`, ktorým zaistíme, aby nás MySQL vyzval k zadaniu hesla. Za týmto parameterom môžme napísať heslo, ktoré sme zadali pri inštalácii MySQL, avšak písať heslo do príkazového riadka nie je príliš bezpečné. Ďalej v príkazovom riadku MySQL servera vytvoríme spomínanú databázu a povolíme do nej prístup užívateľovi `radius` s heslom `radpass`. V reálnom prostredí je nevyhnutné použiť iné, bezpečnejšie heslo, no pre naše testovacie účely postačí aj toto heslo. Použijeme na to tieto príkazy: [28]

```
$ mysql -u root -p
mysql> CREATE DATABASE radius;
mysql> GRANT ALL ON radius.* TO radius@localhost IDENTIFIED BY "
   radpass";
mysql> exit
```

Ďalším krokom je importovanie schémy MySQL databázy. V tejto schéme sú obsiahnuté tabuľky a ich parametre, ktoré využíva FreeRADIUS. Táto schéma ja umiestnená v adresári s pomerne dlho cestou, preto ju uvediem iba v príkaze nižšie. Pre import znova využijeme príkazu `mysql`, v ktorom pomocou znaku < a cesty k schéme, definujeme ktorá schéma sa má importovať. Celý príkaz bude vyzerať následovne: [29]

```
$ mysql -u root -p radius < /etc/freeradius/mods-config/sql/main/
   mysql/schema.sql
```

Po importovaní schémy môžme do databázy `radius`, do tabuľky `radcheck`, ktorá slúži na ukladanie užívateľov, pridať testovacieho užívateľa *bob*. Pri použití tohoto užívateľa je potrebné ho opäť zakomentovať v konfiguračnom súbore FreeRADIUSu `users`. Po tomto kroku môžme pridať užívateľa do databázy MySQL. Využijeme na to príkazový riadok MySQL. Ďalšou výhodou použitia MySQL databázy je, že má v sebe implementovaných množstvo nástrojov, medzi ktoré patrí aj nástroj na vytváranie MD5 kontrolných súčtov. Nemusíme teda heslá ukladať v textovej forme, ale môžeme jednoducho pomocou MD5 súčtu. Užívateľovi bob priradíme heslo *ahoj*, ktoré uložíme v podobe MD5 súčtu. Následne si môžme pomocou príkazu `select * from radcheck;` zobraziť obsah tabuľky `radcheck`, v ktorej by sa mal nachádzať náš užívateľ *bob*. Výstup tohoto príkazu je zobrazený na obrázku 26. Použijeme na to tieto príkazy: [1, 29]

```
$ mysql -u root -p radius
mysql> INSERT INTO radcheck (username, attribute, op, value) VALUES (
   'bob', 'MD5-Password', ':=', MD5( 'ahoj' ));
mysql> select * from radcheck;
mysql> exit
```

```
mysql> select * from radcheck;
+----+----------+--------------+----+----------------------------------+
| id | username | attribute    | op | value                            |
+----+----------+--------------+----+----------------------------------+
|  1 | bob      | MD5-Password | := | 79c2b46ce2594ecbcb5b73e928345492 |
+----+----------+--------------+----+----------------------------------+
1 row in set (0.00 sec)
```

Obr. 26: Výpis užívateľov z tabuľky `radcheck` s uloženým MD5 heslom.

Ďalším krokom je aktivácia modulu `sql` na FreeRADIUSe. Použijeme na to symbolický odkaz podobne ako pri povoľovaní autentifikácie pomocou PAM modulu. Následne v tomto module nastavíme parametre pre prihlasovanie FreeRADIUSu na náš MySQL server. Nájdeme v ňom riadky, ktoré sú uvedené nižšie a upravíme ich podľa nášho nastavenia MySQL. Vykonáme to týmito príkazmi a úpravou súboru:

```
$ sudo ln -s /etc/freeradius/mods-available/sql /etc/freeradius/mods-
   enabled/sql
$ sudo nano /etc/freeradius/mods-enabled/sql
...
      driver = "rlm_sql_mysql"
      server = "localhost"
      port = 3306
      login = "radius"
      password = "radpass"
...
```

Teraz už stačí iba reštartovať službu `freeradius` a môžme sa pustiť do overenia konfigurácie. Na overenie stačí použiť nástroj `raddtest`, v ktorom ako meno použijeme *bob* a ako heslo *ahoj*. V prípade, že bola konfigurácia úspešná mali by sme dostať odpoveď Access-Accept. Príklad úspešnej autentifikácie je zobraný na obrázku 27.

```
student@client:~$ radtest -6 bob ahoj radius.raddp.cz 0 testing123
Sent Access-Request Id 82 from [::]:47310 to [2001:718:1001:2c6::211]:1812 length 85
        User-Name = "bob"
        User-Password = "ahoj"
        NAS-IPv6-Address = 2001:718:1001:2c8:76d4:35ff:fe7c:ece
        NAS-Port = 0
        Message-Authenticator = 0x00
        Cleartext-Password = "ahoj"
Received Access-Accept Id 82 from [2001:718:1001:2c6::211]:1812 to [::]:0 length 20
```

Obr. 27: Úspešná autentifikácia po konfigurácii MySQL databázy.

# 6 Overenie navrhnutého systému v laboratóriu

Overovanie systému prebiehalo v laboratórnych podmienkach, v učebni katedry telekomunikačnej techniky. V tejto učebni sú počítače s operačným systémom Ubuntu 14.04 LTS. Počítačová sieť v učebni pracuje iba na IPv6 adresách, tak isto ako virtuálny server, na ktorom je spustený FreeRADIUS server, MySQL databáza, DNS a NTP server. Všetky konfigurácie prebiehali podľa návodov v kapitole 5. Na klientskej stanici sa líšila iba konfigurácia PAM modulov, aby bol navrhnutý autentifikačný systém implementovaný v celom OS.

## 6.1 Konfigurácia PAM modulov

Pri niektorých činnostiach nie je možné použiť autentifikáciu odtlačkom prsta, a pri niektorých je zbytočné požadovať od užívateľa opakovanú autentifikáciu všetkými tromi metódami. Preto je na mieste konfigurácia určitých aplikácii, pri ktorých nie je potrené alebo možné prejsť celým trojfaktorovým autentifikačným procesom. Všetky konfiguračné súbory aplikácii využívajúcich PAM moduly sa nachádzajú v adresári `/etc/pam.d/`. Typicky príkladom, kedy nie je možné použiť biometrickú autentifikáciu, je pri vzdialenej správe pomocou SSH. Preto upravíme súbor `sshd`, v ktorom sú definované PAM moduly použité pri autentifikácii cez SSH. Otvoríme teda tento súbor a pred riadok, ktorý obsahuje `@include common-auth` vložíme riadky, ktorými definujeme použitie PAM modulu na autentifikáciu cez RADIUS server a PAM modul pre Google Authenticator. Upravená časť by mala vyzerať takto:

```
# Standard Un*x authentication.
auth requisite pam_radius_auth.so
auth sufficient pam_google_authenticator.so echo_verification_code
   nullok
@include common-auth
```

Ďalej je treba ešte povoliť v nastaveniach SSH servera zobrazovanie výziev z PAM modulov. Otvoríme konfiguračný súbor SSH a upravíme riadok, ktorý obsahuje `ChallengeResponseAuthentication no`. Po úprave je potrebné reštartovať službu SSH. Vykonáme to týmito príkazmi a úpravou súboru:

```
$ sudo nano /etc/ssh/sshd_config
...
ChallengeResponseAuthentication yes
...
$ sudo service ssh restart
```

Ako ďalší program, ktorému som upravil nastavenie PAM, bol `sudo`. Pomocou tohoto programu môže užívateľ, ktorý patrí do skupiny `sudo`, spúšťať príkazy ako `root`. Pri vyvolaní tohoto príkazu v defaultnom nastavení, musí užívateľ zadať svoje prihlasovacie heslo.

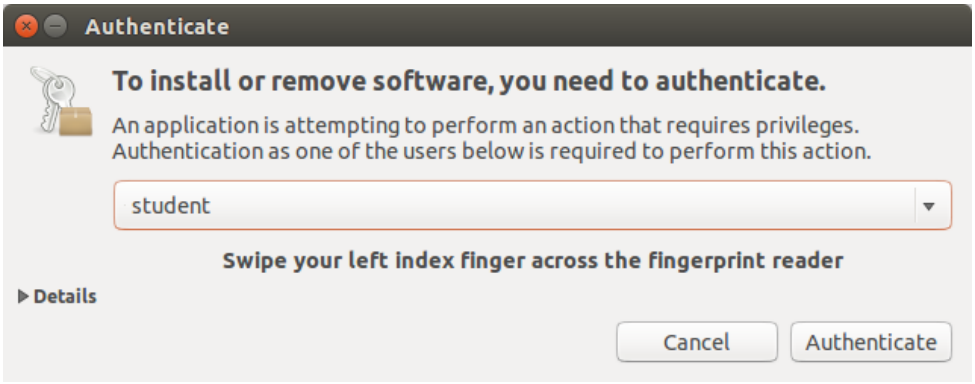
Pre lepšie zabezpečenie, so zachovaním užívateľského komfortu, som nastavil ako dostačujúce overenie PAM modulom na autentifikáciu odtlačkom prsta. V prípade, že užívateľ pristupuje vzdialene pomocou SSH, ako druhý modul som nastavil Google Authenticator, keďže čítačku odtlačkov nie je možné použiť vzdialene. Taktiež som čítačke nastavil čas na vypršanie overenia na 10 sekúnd, aby správca pri vzdialenej správe nemusel čakať pôvodných 30 sekúnd, kým ho systém vyzve na zadanie verifikačného kódu z Google Authenticatora. Otvoril som súbor `sudo` a upravil ho následovne:

```
$ sudo nano /etc/pam.d/sudo
...
auth sufficient pam_fprintd.so timeout=10
auth sufficient pam_google_authenticator.so echo_verification_code
   nullok
@include common-auth
...
```

Ako ďalší som nastavoval PAM moduly pre program `polkit`. Ten poskytuje prístup k privilegovaným operáciám pre neprivilegované aplikácie. Využíva sa napríklad pri inštalovaní aplikácii pomocou programu Ubuntu Software Center, kedy pred inštaláciou zobrazí autentifikačné okno, do ktorého je treba zadať požadované údaje. Konfigurácia je podobná ako pri programe `sudo`, pretože opäť bude stačiť užívateľovi autentifikovať sa pomocou odtlačku prsta a v prípade vypršania času overenia, bude vyzvaný na zadanie OTP hesla. Ukážka autentifikačného okna, ktoré sa zobrazí pri inštalácii nových aplikácii je na obrázku 28. Konfiguráciu vykonáme takto:

```
$ sudo nano /etc/pam.d/polkit-1
...
auth sufficient pam_fprintd.so
auth sufficient pam_google_authenticator.so echo_verification_code
   nullok
@include common-auth
...
```

Ako posledný som upravoval súbor `common-auth`, pomocou ktorého prebieha autentifikácia všetkých programov, ktoré nemajú definovanú inú autentifikáciu. Preto bolo potrebné v predošlých prípadoch umiestniť požadované PAM moduly pred integráciu tohoto konfiguračného súboru. Nastavené PAM moduly sa tak budu používať napríklad pri prihlasovaní užívateľa do systému alebo jeho návratu do systému z uzamknutej obrazovky. Nami požadované PAM moduly je treba umiestniť pred riadok, ktorý obsahuje PAM modul `pam_unix.so`. Ten na autentifikáciu využíva heslo uložené v systéme. Pomocou neho sa môžme prihlásiť do systému v prípade poranenia prsta užívateľa, prípadne správcu, ktorý má nastavené aj ne-

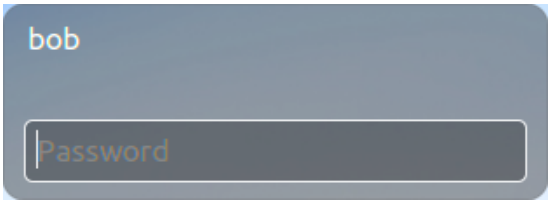



Obr. 28: Autentifikačné okno vyvolané programom `polkit`.

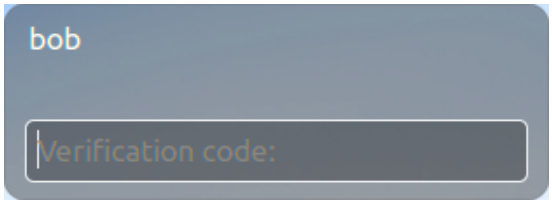
jaké systémové heslo, mimo toho uloženého v MySQL databáze. Konfiguráciu spomínaného súboru `common-auth` vykonáme takto:

```
$ sudo nano /etc/pam.d/common-auth
...
auth requisite pam_radius_auth.so
auth requisite pam_google_authenticator.so echo_verification_code
   nullok
auth sufficient pam_fprintd.so
auth [success=1 default=ignore] pam_unix.so nullok_secure
   try_first_pass
...
```

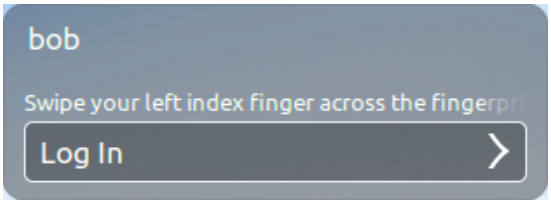
Pri takomto nastavení najprv systém vyzve užívateľa na zadanie hesla, ktoré sa overí na RADIUS serveri. Po úspešnom overení bude vyzvaný na zadanie verifikačného kódu z mobilnej aplikácie Google Authenticator a ak aj ten zadá správne, už mu len zostáva autentifikovať sa pomocou odtlačku prsta. Po úspešnom overení odtlačku, je ešte potrebné kliknúť na tlačítko Log In, pomocou ktorého sa prihlási do systému. Celý tento prihlasovací proces zobrazuje obrázok 29. Pri mojom testovaní som mal zapnutý debugovací mód pri všetkých PAM moduloch a výpisy z logovacieho súboru `auth.log`, spolu s výpisom z logovacieho súboru `radius.log` zo servera, sú v prílohe D.




(a) Overenie hesla voči RADIUS serveru.




(b) Zadanie verifikačného kódu z Google Authenticatora.




(c) Overenie odtlačku prsta a tlačítko Log In.

Obr. 29: Priebeh prihlasovania do systému s navrhnutou trojfaktorovou autentifikáciou.

# 7 Záver

Cieľom tejto práce bolo navrhnúť riešenie s viacfaktorovou autentifikáciou pomocou protokolu RADIUS a biometrie v IPv6 sieťach. V mojom návrhu som použil konkrétne trojfaktorovú autentifikáciu. Keďže som pri práci používal operačný systém Ubuntu, všetky prvky autentifikácie boli implementované pomocou PAM modulov.

Ako RADIUS server, pomocou ktorého sa užívateľ vzdialene autentifikuje, som použil open source riešenie s názvom FreeRADIUS. Na uloženie užívateľských účtov som využil MySQL databázu, ktorú FreeRADIUS server využíva pri autentifikácii užívateľov. Keďže RADIUS protokol využíva pri svojej činnosti doménové mená, je v práci popísaná aj konfigurácia DNS servera.

Biometriu v navrhnutom systéme zastupuje autentifikácia pomocou odtlačku prsta. Existujú však spôsoby ako prelomiť ochranu heslom (napr. útokom hrubou silou) a taktiež je možné relatívne ľahko oklamať použitú čítačku odtlačkov prstov (napr. vytvorením falošného prsta). Preto som sa rozhodol použiť ešte ako dodatočnú ochranu aplikáciu Google Authenticator, pomocou ktorej musí užívateľ pri prihlasovaní zadať jednorázové heslo. Tým zamedzíme možnosti útoku hrubou silou, keďže OTP heslo je menené každých 30 sekúnd. Dá sa povedať, že tento prvok je zo všetkých troch použitých autentifikačných metód najbezpečnejší. Na to aby boli synchronizované OTP heslá v systéme, aj v mobilnej aplikácii, som využil NTP server nainštalovaný na rovnakom serveri ako FreeRADIUS.

Aby však užívateľ takto zabezpečeného systému nemusel pri každej činnosti vyžadujúcej autentifikáciu prejsť znovu celým trojfaktorovým overovaním, bola pri bežných činnostiach nastavená ako dostačujúca autentifikácia odtlačkom prsta. Ďalej bola ako záložná autentifikácia pri takýchto činnostiach nastavená autentifikácia jednorázovým heslom. Tá bola nastavená aj v prípade vzdialenej správy cez SSH, pri ktorej nie je možné použiť čítačku odtlačkov prstov.

V prípade, že si užívateľ poraní prst alebo inak znemožní autentifikáciu odtlačkom prsta, môže byť v systéme definovaný správcovský účet, ktorý bude mať nastavené aj systémové heslo, a teda v prípade neúspešnej autentifikácie odtlačkom prsta použije na prihlásenie toto heslo. Následne môže užívateľovi nasnímať nový odtlačok iného, neporaneného prsta.

Prínosom tejto práce je podrobne popísaná konfigurácia navrhnutého autentifikačného systému s využitím aktuálnych verzii použitých programov v prostredí IPv6 sietí. Konfigurácia je síce popísaná pre operačný systém Ubuntu 14.04, no je ju možné s malými rozdielmi (predovšetkým pri inštalácii softwaru z repozitárov) aplikovať do akejkoľvek Linuxovej distribúcie.

# Literatúra


[1] VAN DER WALT, Dirk. *FreeRADIUS beginner's guide: manage your network resources with FreeRADIUS*. Birmingham [U.K.]: Packt Pub. Ltd., 2011, xiii, 317 p. ISBN 978-1849514088.

[2] O'GORMAN, Lawrence. *Securing Business's Front Door – Password, Token, and Biometric Authentication* [online]. 2004, 2015-11-23 [cit. 2015-11-23]. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.198.6881&rep=rep1&type=pdf

[3] Single sign-on with one time password. *2009 First Asian Himalayas International Conference on Internet* [online]. IEEE, 2009, : 1-4 [cit. 2015-11-29]. DOI: 10.1109/AHICI.2009.5340290. ISBN 978-1-4244-4569-1. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5340290

[4] MENEZES, Alfred, Paul C VAN OORSCHOT a Scott A VANSTONE. *Handbook of applied cryptography*. Boca Raton: CRC Press, c1997, xiii, 780 s. Discrete mathematics and its applications. ISBN 08-493-8523-7. Dostupné z: http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.99.2838&rep=rep1&type=pdf

[5] TAEKYOUNG KWON a JIN HONG. Analysis and Improvement of a PIN-Entry Method Resilient to Shoulder-Surfing and Recording Attacks. *IEEE Transactions on Information Forensics and Security* [online]. 2015, extbf10(2): 278-292 [cit. 2015-12-05]. DOI: 10.1109/TIFS.2014.2374352. ISSN 15566013. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6966749

[6] MEYER, Roger. *Secure Authentication on the Internet* [online]. 2007 [cit. 2015-12-05]. Dostupné z: https://www.sans.org/reading-room/whitepapers/securecode/secure-authentication-internet-2084

[7] *RSA SecurID Software Tokens* [online]. [cit. 2015-12-05]. Dostupné z: http://www.emc.com/security/rsa-securid/rsa-securid-software-tokens.htm

[8] *Domain Name Service (DNS)* [online]. [cit. 2016-03-30]. Dostupné z: https://help.ubuntu.com/lts/serverguide/dns-configuration.htm

[9] *How do I setup a reverse DNS?* [online]. [cit. 2016-03-30]. Dostupné z: https://www.sixxs.net/faq/dns/?faq=reverse

[10] *Common Record Types* [online]. [cit. 2016-04-01]. Dostupné z: https://help.ubuntu.com/lts/serverguide/dns-references.html#dns-more-info

[11] *Troubleshooting* [online]. [cit. 2016-04-01]. Dostupné z: https://help.ubuntu.com/lts/serverguide/dns-troubleshooting.html

[12] *Network Manager* [online]. [cit. 2016-04-01]. Dostupné z: https://help.ubuntu.com/community/NetworkManager

[13] *Dnsmasq* [online]. [cit. 2016-04-04]. Dostupné z: http://www.thekelleys.org.uk/dnsmasq/doc.html

[14] *IPv6 address by hostname [closed]* [online]. [cit. 2016-04-06]. Dostupné z: https://stackoverflow.com/questions/16467765/ipv6-address-by-hostname

[15] *CONFIGURING HOST NAMES USING HOSTNAMECTL* [online]. [cit. 2016-04-06]. Dostupné z: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Networking_Guide/sec_Configuring_Host_Names_Using_hostnamectl.html

[16] *12 scp command examples to transfer files on Linux* [online]. [cit. 2016-04-06]. Dostupné z: http://www.binarytides.com/linux-scp-command/

[17] *Setup SSH to Authenticate off FreeRADIUS on CentOS 6.3 + Ubuntu* [online]. 2012 [cit. 2016-04-08]. Dostupné z: http://safesrv.net/setup-ssh-to-authenticate-off-freeradius-on-centos-6-3/

[18] *Configuring Radius Authentication on Linux* [online]. 2014 [cit. 2016-04-08]. Dostupné z: http://mikedixson.com/2014/09/configuring-radius-authentication-on-linux/

[19] *FreeRADIUS Google Dual Factor Authenticator* [online]. [cit. 2016-04-08]. Dostupné z: http://www.supertechguy.com/help/security/freeradius-google-auth

[20] *FreeRADIUS Documentation - rlm_pam* [online]. [cit. 2016-04-08]. Dostupné z: http://networkradius.com/doc/3.0.10/raddb/mods-available/pam.html

[21] *Wiki - Home* [online]. In: . [cit. 2016-04-09]. Dostupné z: https://github.com/google/google-authenticator/wiki

[22] *Google Authenticator PAM module* [online]. [cit. 2016-04-09]. Dostupné z: https://github.com/google/google-authenticator/blob/master/libpam/README.md

[23] *Globally distributed IPv4 and IPv6 NTP service* [online]. [cit. 2016-04-09]. Dostupné z: https://www.sixxs.net/tools/ntp/

[24] *Configure NTP Server (NTPd)* [online]. [cit. 2016-04-09]. Dostupné z: http://www.server-world.info/en/note?os=Ubuntu_14.04&p=ntp

[25] *FreeRADIUS Google Dual Factor Authenticator* [online]. [cit. 2016-04-09]. Dostupné z: http://www.supertechguy.com/help/security/freeradius-google-auth

[26] *How to enable integrated fingerprint reader with fprint* [online]. [cit. 2016-04-10]. Dostupné z: http://www.thinkwiki.org/wiki/How_to_enable_integrated_fingerprint_reader_with_fprint

[27] *Fprint* [online]. [cit. 2016-04-10]. Dostupné z: https://www.freedesktop.org/wiki/Software/fprint/

[28] *Guide/SQL HOWTO* [online]. [cit. 2016-04-10]. Dostupné z: http://wiki.freeradius.org/guide/SQL-HOWTO

[29] Hogan, M. Are you who you claim to be ? *International Standards Organisation: National Institute of Standards and Technology* [online]. 2003 [cit. 2016-04-15]. Dostupné z: http://www.iso.org/iso/livelinkgetfile?llNodeId=20993&llVolId=-2000

[30] Ščurek, Radomír *Biometrické metody identifikace osob v bezpečnostní praxi* [online]. 2008 [cit. 2016-04-15]. Dostupné z: https://www.fbi.vsb.cz/export/sites/fbi/040/.content/sys-cs/resource/PDF/biometricke_metody.pdf

[31] KHOURY, Franjieh El. *Iris biometric model for secured network access*. Boca Raton: CRC Press, 2013. ISBN 978-146-6502-130.

[32] BENEŠ, R. *Autentizační metody založené na biometrických informacích* [online]. 2010 [cit. 2016-04-15]. Dostupné z: http://access.feld.cvut.cz/view.php?cisloclanku=2010110002

[33] GREGORY, Peter H a Michael A SIMON. *Biometrics for dummies*. Indianapolis, IN: Wiley Publishing, Inc., 2008. ISBN 04-702-9288-1.

[34] Fingerprint Recognition. *NSTC Subcommittee on Biometrics* [online]. 2006 [cit. 2016-04-15]. Dostupné z: http://www.biometrics.gov/Documents/FingerprintRec.pdf

[35] JAIN, Anil K, Arun A ROSS a Karthik NANDAKUMAR. *Introduction to biometrics*. New York: Springer, c2011. ISBN 978-038-7773-261.

[36] JAIN, Anil K, Patrick FLYNN a Arun A ROSS. *Handbook of biometrics*. New York: Springer, c2008. ISBN 978-038-7710-419.

[37] Face Recognition. *NSTC Subcommittee on Biometrics* [online]. 2006 [cit. 2016-04-16]. Dostupné z: http://www.biometrics.gov/Documents/FaceRec.pdf

[38] Speaker Recognition. *NSTC Subcommittee on Biometrics* [online]. 2006 [cit. 2016-04-16]. Dostupné z: http://www.biometrics.gov/Documents/SpeakerRec.pdf

[39] MUSHTAQ, Saba a A. H. MIR. Signature verification: A study. *2013 4th International Conference on Computer and Communication Technology (ICCCT)* [online]. IEEE, 2013, , 258-263 [cit. 2016-04-18]. DOI: 10.1109/ICCCT.2013.6749637. ISBN 978-1-4799-1572-9. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6749637

[40] *Remote Authentication Dial In User Service (RADIUS).* C. Rigney, S.Willens, A. Rubens, W. Simpson. [online]. 2000 [cit. 2016-04-19]. Dostupné z: https://tools.ietf.org/html/rfc2865

[41] *THE FREERADIUS TECHNICAL GUIDE* [online]. 2014 [cit. 2016-04-19]. Dostupné z: http://networkradius.com/doc/FreeRADIUS%20Technical%20Guide.pdf

[42] *About the FreeRADIUS Project* [online]. [cit. 2016-04-20]. Dostupné z: http://freeradius.org/about.html

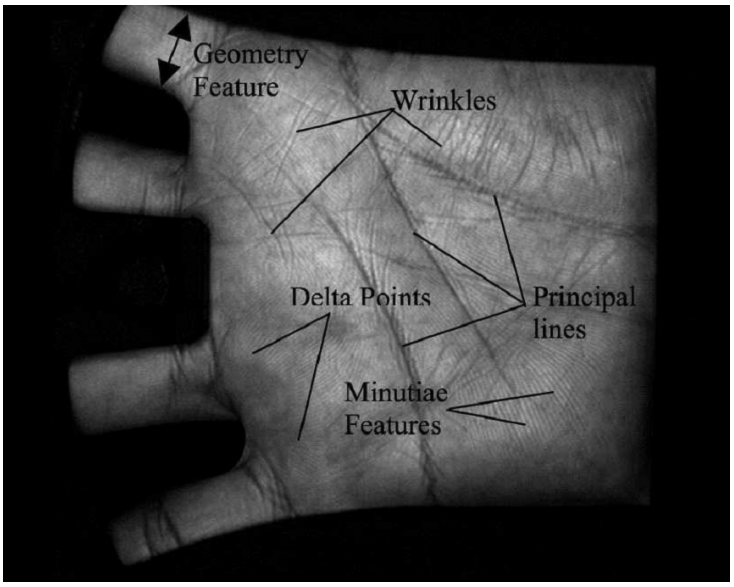
[43] MORGAN, Andrew G. a Thorsten KUKUK. *The Linux-PAM System Administrators' Guide* [online]. 2010 [cit. 2016-04-20]. Dostupné z: http://www.linux-pam.org/Linux-PAM-html/Linux-PAM_SAG.html

# A Biometrická autentifikácia

## A.1 Fyziologické metódy

### A.1.1 Odtlačok dlane

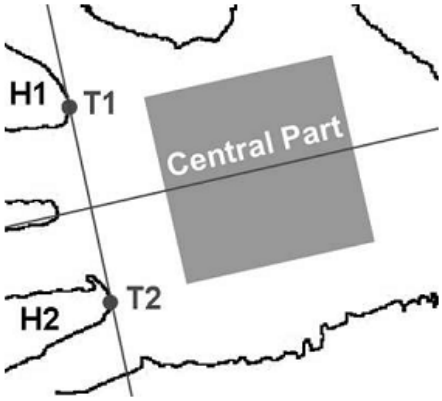
Za odtlačok dlane je považovaná vnútorná strana ruky. Povrch dlane je pokrytý rovnakým druhom pokožky ako špičky prstov, pričom má však podstatne väčšie rozmery. Preto sa pri identifikácii na základe odtlačku dlane využívajú rovnaké vlastnosti ako pri odtlačku prsta, ktoré sú doplnené o niektoré ďalšie. Sú tu využité geometrické vlastnosti, ktoré nám vďaka tvaru dlane umožňujú pomerne jednoducho určiť jej šírku, výšku a obsah. Hlavné línie, ktoré sa menia v čase iba veľmi málo, a preto zohrávajú podstatnú úlohu pri rozpoznávaní dlane. Ďalej sa na dlani vyskytujú vrásky, ktoré sa od hlavných línii líšia predovšetkým tým, že sú o niečo tenšie a ich výskyt je nepravidelný. Na dlani sa taktiež nachádzajú rovnaké singularity ako na prste, a to v tvare oblúka, slučky, víru a delty. Delty sa tu vyskytujú najčastejšie a je možné ich pozorovať pri koreni prstov. Rovnako ako na prste sa aj tu skúmajú markanty v podobe zakončenia papilárnych línii. Všetky skúmané vlastnosti na dlani zobrazuje obrázok 30. [36]




Obr. 30: Vlastnosti skúmané pri identifikácii na základe odtlačku dlane. [36]

Zvyčajne sa systém na rozpoznanie dlane skladá z troch základných častí. Prvou časťou je užívateľské rozhranie, ktoré komunikuje s užívateľom pre správne zosnímanie dlane. Druhá časť slúži na samotné získavanie obrázkov dlane, a to pomocou priehľadného plochého povrchu, na ktorý sa položí dlaň a CCD kamery, ktorá ju zosníma. Najdôležitejšou časťou je rozpoznávací modul, ktorý rozhoduje o tom či bude alebo nebude užívateľ úspešne autentifikovaný. Má za úlohu predbežné spracovanie snímku ruky, ktorým zabezpečíme, aby z dlaní rôznej

veľkosti zosnímaných pod rozdielnym uhlom, bola vyňatá iba ich stredná časť. Túto časť zobrazuje obrázok 31. Z nej sú potom vybrané charakteristické vlastnosti, ktoré sa využívajú pri samotnej autentifikácii. [36]




Obr. 31: Centrálna časť dlane využívaná pri autentifikácii. [36]

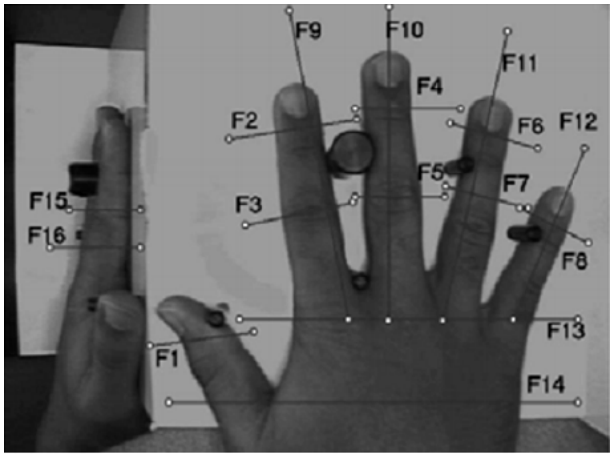
Tento systém predstavuje strednú cestu medzi systémami využívajúcimi odtlačok prsta a geometriu ruky. Hodnota FRR má hodnotu $0.86\%$, čo je o niečo lepšie ako pri odtlačku prsta, ale horšie ako pri rozpoznávaní pomocou geometrie ruky. Podobne na tom je aj koeficient FRR s hodnotou $0.017\%$.

### A.1.2 Geometria ruky

Geometria ruky ako autentifikačná metóda ma dlhú históriu používania. Prvý komerčný skener pre geometriu ruky mal názov Identimat a bol predstavený na začiatku sedemdesiatych rokov minulého storočia. Pri skenovaní sa využívala 1000 wattová žiarovka, ktorá aktivovala mechanické skenovanie pomocou fotobuniek na meranie tvaru ruky. V roku 1986 firma Recognition Systems prišla na trh so skenermi, ktoré pomocou CCD kamery zachytávajú 3D obraz ruky a ukladajú ho do malého 9 bytového súboru, čo činí tento spôsob autentifikácie šetrný z pohľadu požadovaného pamäťového priestoru. Všeobecne sa geometria ruky využíva hlavne k získaniu fyzického prístupu do stráženého objektu, kde sa pohybuje veľké množstvo ľudí ako napríklad letiská. V USA je často nasadzovaná na autentifikáciu pri vstupe do jadrových elektrární. Veľkosť ruky sa mení spolu s narastajúcim vekom, pričom najviac pri vývine, a preto nie je táto metóda vhodná pre autentifikáciu detí. Taktiež nie je možné identifikovať ľudí, ktorým chýba niektorý z prstov na ruke. [31, 36]

Väčšina systémov na geometriu ruky vyžaduje obrázok jej zadnej strany. Preto je potrebné pri skenovaní umiestniť ruku na skener dlaňou dole. Na zachytenie obrázka ruky sa najčastejšie využíva systém, ktorý je zložený z rovného povrchu na ktorý je potrebné umiestniť ruku, z kolíkov, pomocou ktorých zaistíme správne umiestnenie ruky a z CCD kamery, ktorá zachytí výsledný obrázok. Z tohoto obrázku sa následne získavajú typické vlastnosti ruky. Jedná

sa o geometrické vlastnosti, medzi ktoré sa radia dĺžka a šírka prstov, dĺžka a šírka dlane a hrúbka prstov. Meranie týchto vlastností je graficky zobrazené na obrázku 32. [35]




Obr. 32: Proces merania typických vlastností ruky. [35]

Pri tomto systéme sa hodnota koeficientu FRR pohybuje na hodnote <0.1% a hodnota koeficientu FAR je 0.1%. Hodnota FAR je omnoho vyššia ako pri použití odtlačku prsta, čo činí tento systém iba stredne spoľahlivý. K verifikácie dôjde pomerne rýchlo, v čase $1-2$ sekundy. [30]

### A.1.3 Rozpoznávanie tváre

Systémy na rozpoznanie tváre sú veľmi žiadané v boji proti trestnej činnosti. Taktiež sú aplikované pri autentifikácii užívateľov pre získanie prístupu do fyzického alebo virtuálneho priestoru. Avšak úloha identifikácie osoby tým, že sa vezme vstupný obrázok tváre a porovná sa so snímkami uloženými v databáze, je stále veľmi náročný problém. Je to spôsobené premenlivosťou ľudskej tváre pri rôznych druhoch podnetov, ako napríklad osvetlenie, výraz tváre, uhol pohľadu na tvár, starnutie alebo aj nosenie okuliarov. Často má zmena týchto podnetov výrazný vplyv na výkonnosť systému na rozpoznávanie tváre, a to obzvlášť ak ju má porovnávať s rozsiahlou databázou tvári. Tento problém bráni širokému nasadeniu takýchto systémov v reálnych podmienkach. Napríklad v roku 2002 prebehol živý test rozpoznávania tváre na medzinárodnom letisku Palm Beach. Pri rozpoznávaní tvári zamestnancov, ktorý sa dobrovoľne prihlásili a ich tváre boli nasnímané do databázy, zlyhal tento systém v 53% prípadoch. Týchto 250 zamestnancov mal systém rozpoznať z pomedzi 5000 cestujúcich, pričom však spúšťal falošný poplach dva až tri krát za hodinu. Rozpoznávanie tváre je však vhodné pri kontrolovaných podmienkach, keď porovnávané snímky tváre sú zhotovené za rovnakých podmienok. [33, 36]

Algoritmy na rozpoznávanie tváre využívajú pri svojej činnosti vlastnosti a geometrické vzťahy, ako sú obsah, vzdialenosť a uhly, medzi charakteristickými rysmi tváre. Môžme ich zaradiť do dvoch všeobecných kategórii, v závislosti na tom aké vlastnosti tváre využívajú

na jej reprezentáciu. Prvá z nich je „appearance-based“ technika, pri ktorej je generovaná kompaktná reprezentácia celej oblasti tváre pomocou mapovania obrázku tváre zloženého z viac-dimenzionálneho vektorového priestoru na menej-dimenzionálny vektorový podpriestor. Tento podpriestor je definovaný sadou bázových vektorov, ktoré sa získavajú pri snímaní tváre. Typickými predstaviteľmi tejto techniky sú algoritmy PCA a LDA. Druhá technika sa nazýva „model-based“ a je pri nej vytvorený 2D alebo 3D model tváre, čo uľahčuje rozpoznávanie tváre v prípade ak sa nachádza v rôznych pózach. Najznámejším algoritmom v tejto kategórii je EBGM. [35, 36]

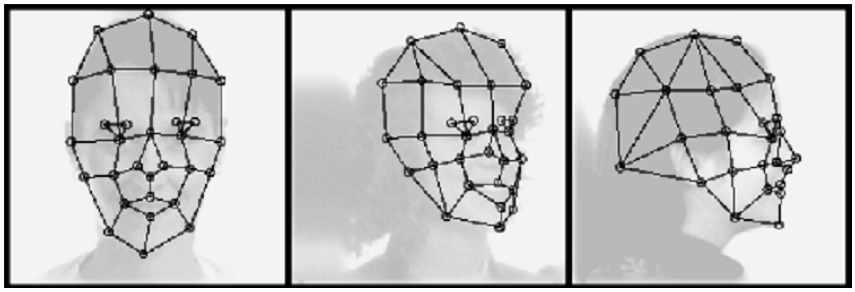
**PCA - Principal Components Analysis** - je to jedna z prvých automatizovaných metód určených na rozpoznávanie tváre. Zo vzorových dát určuje vektorové podpriestory, ktoré zodpovedajú čo najväčšej variabilite vo vzorových dátach. To je dosiahnuté pomocou vyjadrenia vlastných čísiel (Eigenvalue) s najvyššou hodnotou z kovariančnej matice skúmaných dát. Z týchto vektorových podpriestorov sú vytvorené takzvané „Eigenfaces“, ktoré sa používajú pri samotnom rozpoznávaní tváre. Na obrázku 33 sú zobrazené Eigenfaces získané pri snímaní tváre. [35, 36]

Obr. 33: Príklad Eigenfaces získaných pomocou PCA. [36]

**LDA - Linear Discriminant Analysis** - používa triedenie vzorky dát do takzvaných tried a vykonáva analýzu vektorových podpriestorov s cieľom minimalizovať rozdiely v rámci jednej triedy, zatiaľ čo maximalizovať rozdiely medzi rozličnými triedami. Vďaka tomu dokáže poskytnúť LDA presnejšie rozpoznanie tváre, za predpokladu, že bol zosnímaný dostatok vzoriek tvári jednotlivých užívateľoch. [35]

**EBGM - Elastic Bunch Graph Matching** - táto metóda, narozdiel od predchádzajúcich, využíva toho, že na tvári je mnoho nelineárnych charakteristík, ktoré nie sú ovplyvnené podnetmi spomenutými na začiatku tejto kapitoly. Pomocou Gaborovej vlnkovej transformácie sú vytvorené dynamické spojenia, ktoré sa premietajú na tvár v podobe elastickej mriežky. Takáto mriežka je zobrazená na obrázku 34. Pri tejto metóde sa vyžaduje presná lokalizácia dôležitých bodov, a preto je často kombinovaná s metódami PCA a LDA. [37]

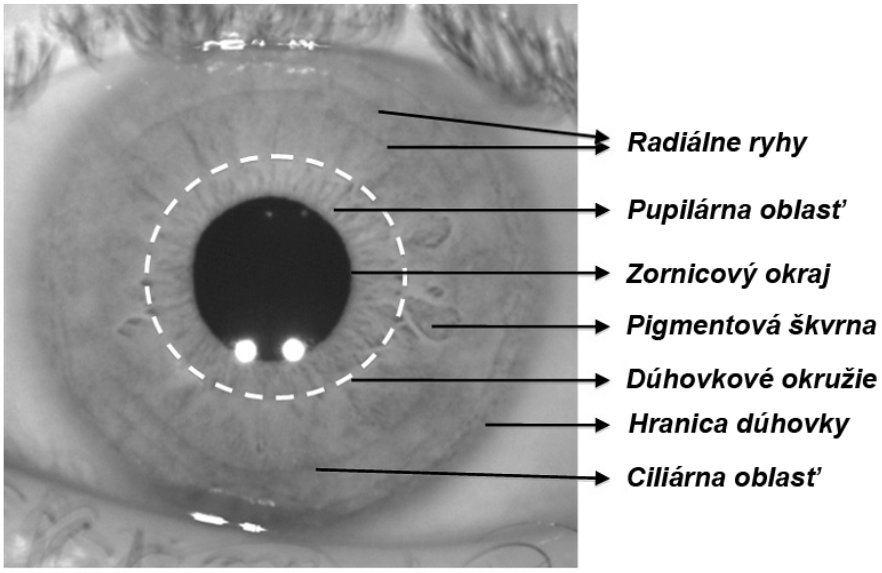
Hodnota FRR je aj pri tejto autentifikačnej metóde nízka a pohybuje sa na hodnotách <1.0%. Koeficient FAR má hodnotu 0.1%. Systém s týmito hodnotami koeficientov môžme označiť ako stredne spoľahlivý. Čas potrebný na autentifikáciu je 3 sekundy. [30]



Obr. 34: Elastická mriežka vytvorená pomocou EBGM. [36]

### A.1.4 Dúhovka

Táto metóda je ešte relatívne mladá, keďže významné pokroky v rozpoznávaní pomocou dúhovky oka, nastali až po roku 1993. Zatiaľ čo rôzne prvky oka boli navrhnuté ako biometrický indikátor, napríklad sietnica alebo spojivky cievok, dúhovka je tá, ktorej boli venované rozsiahle štúdie a je vo veľkej miere nasadzovaná v biometrických systémoch. Dúhovka je vnútorný orgán oka, ktorého primárna funkcia je regulovať množstvo svetla, ktoré vstupuje do oka. Vykonáva to rozširovaním zorničky pri nízkej intenzite svetla a jej zmenšovaním pri vyššej intenzite. Farba dúhovky je závislá na geneticky stanovených vlastnostiach. Tá však pri rozpoznávaní dúhovky nehrá príliš veľkú rolu. Dôležitá je tu textúra dúhovky, ktorá sa tvorí ešte pred narodením a je geneticky nezávislá. Vzor tvorený textúrou vykazuje značné rozdiely naprieč populáciou a dokonca aj identické dvojčatá majú rozdielnu textúru dúhovky. Bohatú štruktúru dúhovky tvorí svalovina, väzivové tkanivo a predná hraničná vrstva. Anatomické vlastnosti dúhovky zobrazuje obrázok 35. [35]

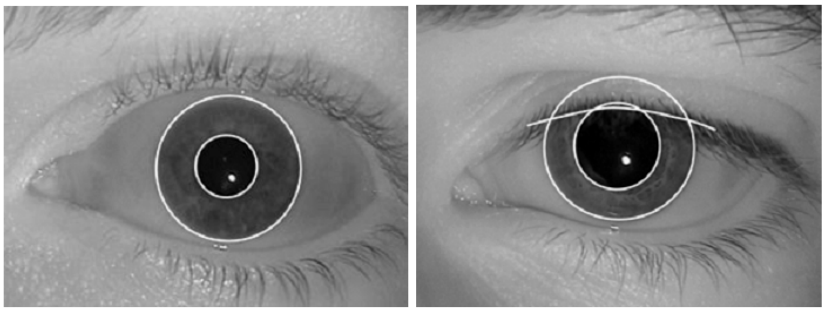



Obr. 35: Anatomické vlastnosti dúhovky zachytené NIR kamerou. [35]

Systémy na rozpoznávanie dúhovky majú za úlohu porovnať dve dúhovky a určiť úroveň ich podobnosti alebo rozličnosti. Takýto systém typicky pozostáva zo štyroch modulov: [35]

1. **Zber dát** - úlohou tohto modulu je získať 2D snímok oka, na čo sa zvyčajne používa monochromatická CCD kamera, ktorá je citlivá na oblasť elektromagnetického spektra pohybujúceho sa v blízkosti infračerveného svetla (NIR). Preto sa na osvetlenie dúhovky používajú infračervené LED diódy, ktoré vyžarujú svetlo s vlnovou dĺžkou rozmedzí 700-900nm. Pri snímaní sietnice je pri väčšine systémov vyžadované aby bolo oko umiestnené v blízkosti snímacej kamery. Pri snímaní sa zachytí väčšie množstvo obrázkov oka, ale systém si vyberie iba tie, u ktorých predpokladá, že majú dostatok informácii o štruktúre dúhovky pre ďalšie spracovanie. Snímky dúhovky môžu byť negatívne ovplyvnené niektorými faktormi, ako napríklad privreté viečko oka, prenikavé alebo nerovnomerné osvetlenie a extrémne rozšírené alebo zmenšené zorničky. Tieto faktory je možné odstrániť regulovaním intenzity osvetlenia a interakciou medzi rozpoznávacím systémom a užívateľom. Ukážkové snímky je možné vidieť na obrázoku 36.

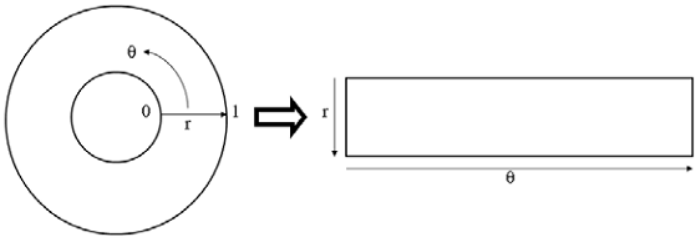
2. **Segmentácia** - pri segmentácii je v obrázku nájdená plocha, ktorá zodpovedá samotnej dúhovke. Vykonáva sa to jej oddelením od ostatných prvkov, ktoré sa tiež nachádzajú na snímku a ide o očné bielko, zorničku, očné viečko a mihalnice. Táto segmentácia je vykonaná pomocou detekcie vnútorných a vonkajších ohraničení dúhovky, konkrétne zornicového okraju a hranice dúhovky. Taktiež detekuje očné viečka a mihalnice, ktoré môžu narušiť inak okrúhlu hranicu dúhovky. Segmentácia dúhovky od ostatných prvkov je kľúčový komponent všetkých biometrických systémov založených na dúhovke. Nesprávne určenie plochy s dúhovkou ma zásadný vplyv na rozpoznanie osoby. Na obrázku 36 je zobrazený výsledok segmentácie dúhovky keď ju nič nezakrýva (na obrázku vľavo) aj keď je dúhovka čiastočne zakrytá očným viečkom (na obrázku vpravo).



Obr. 36: Príklady výstupu segmentácie dúhovky. [35]

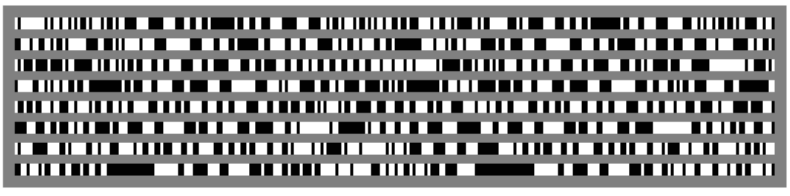
3. **Normalizácia** - po vykonaní segmentácie nasleduje proces normalizácie. Tento proces býva označovaný ako rozbaľovanie dúhovky, pretože jeho výsledkom je obdĺžnikový obrázok, pričom riadky tohoto obdĺžnika predstavujú uhlovú vzdialenosť z pôvodnej okrúhlej dúhovky a stĺpce vyjadrujú vzdialenosť medzi hranicami dúhovky. Pre lepšiu

predstavu je tento proces zobrazený na obrázku 37. Dôvodov prečo je potrebné vykonať normalizáciu dúhovky je hneď niekoľko. Veľkosť zorničky je rôzna naprieč populáciou, a tým pádom aj plocha zodpovedajúca dúhovke je rozdielna. Pomocou normalizácie zaistíme, že dúhovka bude mapovaná do obrázka rovnakých rozmerov. Ďalej to odstraňuje problém rozpoznania dúhovky, keď užívateľ nepatrne nakloní hlavu pri jej snímaní. Každá rozbalená dúhovka je spojená s binárnou maskou, ktorá označuje pixely, na ktorých sa nachádza dúhovka, logickou „1“. Tým ich odlišuje od pixelov, na ktorých sú zobrazené očné viečka a mihalnice, ktoré značí logickou „0“. Nasleduje geometrická normalizácia, pri ktorej je za pomoci fotometrických transformácii zvýraznená štruktúra textúr rozbalenej dúhovky.




Obr. 37: Proces normalizácie dúhovky. [35]

4. **Kódovanie a porovnávanie** - i keď normalizovaná dúhovka môže byť priamo použitá na porovnanie dvoch dúhoviek, pomocou korelačných filtrov, zvyčajne porovnávaniu predchádza proces kódovania, pri ktorom sú vyňaté typické vlastností zo štruktúry dúhovky. Na kódovanie je najčastejšie využívané kvadratúrne 2D Gaborové vlnkovanie, ktorého výstupom je 2D binárny kód nazývaný „iris kód“. Dva takéto kódy sú porovnané pomocou Hammingovej vzdialenosti, ktorá určuje počet odpovedajúcich bitov, ktoré sú rozdielne v týchto dvoch kódoch. Pri takomto porovnávaní je využitá binárna maska, ktorá bola vypočítaná pri segmentácii dúhovky, a zaisťuje, aby sa porovnávali iba pixely zodpovedajúce dúhovke. Pri počítaní Hammingovej vzdialenosti je potrebné, aby boli porovnávané iris kódy presne zarovnané. Hammingova vzdialenosť rovnakých dúhoviek je nižšia ako dvoch odlišných dúhoviek. Iris kód je často reprezentovaný v piktografickom znázornení a jeho ukážka je zobrazená na obrázku 38.



Obr. 38: Piktografické znázornenie Iris kódu. [36]

Táto metóda autentifikácia je jedna z najspoľahlivejších čo dokazuje aj veľmi nízka hodnota jej koeficientov FRR a FAR. Hodnota FRR je $0.00066\%$ a hodnota FAR je $0.00078\%$. Čas potrebný na verifikáciu je 2 sekundy. [30]

## A.2 Behaviorálne metódy

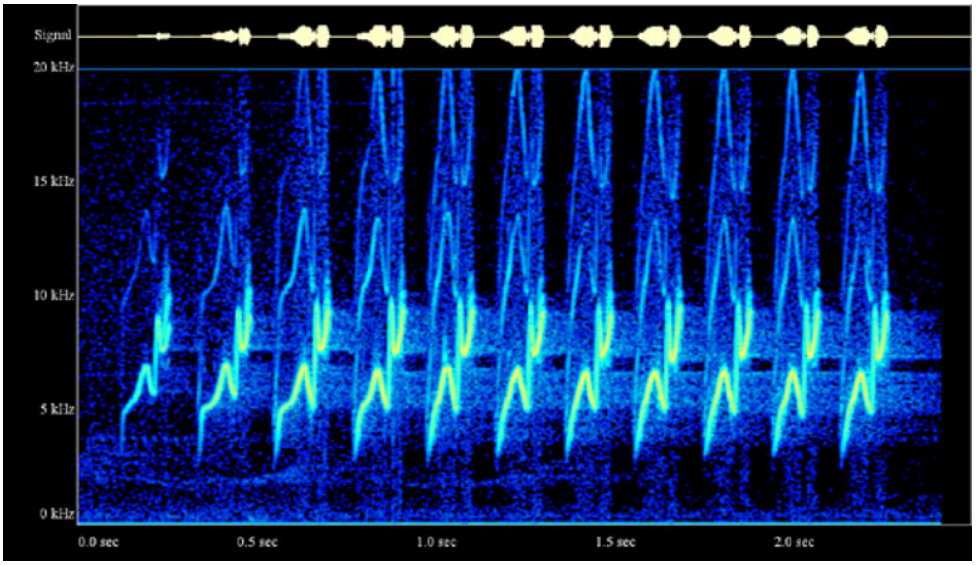
### A.2.1 Hlasová biometria

Pri rozpoznávaní hlasu sa využívajú obidva základné typy biometrie. Fyziologickú časť tu zastupuje vokálny trakt, zložený z hlasiviek, hrtanu, hltanu, jazyka a ústnej dutiny, ktorého špecifický tvar ovplyvňuje farbu hlasu. Behaviorálnu časť predstavuje pohyb úst a spôsob výslovnosti hlások. Biometria hlasu sa využíva na dve základné činnosti. Prvá je autentifikácia na základe hlasu, ktorá slúži na kontrolu prístupu a je zvyčajne overovaná prostredníctvom telefónu. Druhá je na identifikáciu osoby podľa hlasu a býva často nasadzovaná v call centrách, kedy ešte pred prepojením na operátora prebehne identifikácia, na základe ktorej bude operátor hneď vedieť s kým hovorí. Taktiež sa využíva pri odpočúvaní na identifikáciu zaznamenanej osoby. [36, 38]

Hlasový signál má v sebe zakódovaný jazykový obsah a v závislosti na tom ako sa využije sa delia technológie na rozpoznanie hlasu na dva následujúce typy: [36, 38]

- **Obsahovo závislé (*text-depended*)** - pri ktorej musí užívateľ vysloviť špecifickú frázu (napr. „Sezam, otvor sa“) alebo nejakú náhodnú postupnosť („1-2-3“). Táto metóda sa typicky využíva na autentifikáciu užívateľa. Často je kombinovaná so systémom založeným na hesle, kedy užívateľ musí poznať toto heslo a ešte ho aj on sám vysloviť. V takomto systéme je pri získaní vzorových dát užívateľ vyzvaný aby vyslovil slovo alebo frázu, ktorá je potom nahraná pomocou mikrofónu. Následne sú z nahratej frázy vyňaté špecifické vlastnosti hlasu (trvanie, dynamika intenzity hlasu, výška hlasu), z ktorých je vytvorený takzvaný spektrálny model tejto frázy. Ukážka tohoto modelu je zobrazená na obrázku 39. Najčastejšie sa pri tejto technológii používa Hidden Markov Model, ktorý poskytuje štatistickú reprezentáciu zvuku vytvorenú jednotlivcom. Pri porovnávaní vzoriek je vypočítaná pravdepodobnosť, že vstupná vzorka pochádza od tej istej osoby ako zaznamenaný spektrálny model.

- **Obsahovo nezávislé (*text independed*)** - pri tomto type rozpoznávacej technológie nemá systém znalosť o jazykovom obsahu fráz, ktoré rozpoznávaná osoba vyslovuje. Vzhľadom na tento fakt je identifikácia osoby omnoho komplikovanejšia ako pri systéme závislom na obsahu frázy. Preto sa tento systém používa hlavne na identifikáciu osôb. Významné pokroky tento systém zaznamenal v posledných dvoch dekádach. Od roku 1996 vydáva inštitút NIST vyhodnotenie nazvané Speaker Recognition Evaluations, v ktorom zhŕňa novo nadobudnuté poznatky v rozpoznávaní osôb podľa hlasu.




Obr. 39: Ukážka spektrálneho modelu vytvoreného z nahratého hlasu. [38]

I keď je hlasová biometria pomerne jednoduchá na implementáciu, má niekoľko zásadných slabín, ktoré ju robia menej bezpečnou. Hoci je hlas každého človeka jedinečný, niektoré jeho charakteristiky sa menia v závislosti na zdravotnom stave, emočnom rozpoložení a veku danej osoby. Taktiež okolitý šum a kvalita prenosového kanálu môžu zásadne vplývať na výkonnosť takéhoto systému. Naviac je možné tento systém oklamať imitovaním hlasu danej osoby alebo nahraním a následnom prehraní overovacej frázy pri autentifikácii. [31, 38]

### A.2.2 Biometria podpisu

Skúmanie automatického rozpoznania podpisu je veľmi dôležité, pretože ručne písaný podpis je akceptovaný po právnej aj sociálnej stránke. Veľkou výhodou pri využívaní podpisu ako biometrickej metódy je, že ho môžme získať jednoducho pomocou pera a papiera alebo existujúcimi elektronickými prostriedkami ako je tablet, PDA, dotyková obrazovka a iné. Napriek týmto výhodám biometrie podpisu, stále zostávajú niektoré náročné problémy, ktoré treba vyriešiť. Biometria podpisu má malú univerzálnosť, pretože nie každý je schopný sa podpísať. Ďalším problémom je krátka trvácnosť, keďže ručne písaný podpis má tendenciou sa s postupom času meniť a je zraniteľný na útoky sfalšovaním podpisu. Naviac je podpis, rovnako ako hlasová biometria, ovplyvnený náladou, únavou a pod.

Overenie podpisu sa delí na dve základné metódy, v závislosti na dostupných vstupných informáciach o podpise, a to na on-line a off-line. Na overenie podpisu metódou off-line je využitý statický obrázok podpisu, ktorý sa porovnáva s obrázkom podpisu uloženom v databáze. Pri tejto metóde sa podpis porovnáva na základe jeho výšky, šírky a obrysov. Pri on-line rozpoznávaní podpisu ja šanca správneho rozpoznania podstatne vyššia, a to vďaka mnohým dynamickým vlastnostiam podpisu, ktoré nie sú dostupné z jeho statickej reprezentácie. Na

získanie podpisu pri tejto metóde je využité elektronické snímacie zariadenie, ako napríklad dotyková obrazovka alebo tablet. Pomocou tohto zariadenia sú zaznamenané presné súradnice podpisu, z ktorých je možné určiť smer a rýchlosť ťahu pri jeho písaní. Vo väčšine prípadov sa zaznamenáva aj tlak vyvinutý pri písaní podpisu na podložku. Všetky tieto aspekty sa potom využívajú pri overovaní pravosti podpisu. Často sa kombinujú obidve metódy pre zabezpečenie čo najpresnejšej verifikácie.

V praxi sa najčastejšie stretávame s overovaním podpisu pri vykonávaní finančných transakcií v banke. Biometria podpisu je tiež využívaná pri overovaní pravosti podpisu na právnych dokumentoch. Taktiež sa využíva ako autentifikačná metóda pri prihlasovaní do systému, napríklad v mobilnom telefóne či v tablete. [36, 39]

# B Konfiguračné súbory DNS servera

## B.1 named.conf.local

```
//
// Do any local configuration here
//

// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";

zone "raddp.cz" {
      type master;
      file "/etc/bind/db.raddp.cz";
};

zone "c.2.0.1.0.0.1.8.1.7.0.1.0.0.2.ip6.arpa" {
      type master;
      file "/etc/bind/db.2001.718.1001.2c";
};
```

## B.2 db.raddp.cz

```
; Dopredna zona pre domenu raddp.cz.
;
$TTL  86400
@     IN    SOA   raddp.cz. root.raddp.cz. (
                        1       ; Serial
                   604800     ; Refresh
                    86400     ; Retry
                  2419200     ; Expire
                    86400 )   ; Negative Cache TTL
;
@     IN    NS    ns.raddp.cz.
@     IN    AAAA  2001:718:1001:2c6::211
ns    IN    AAAA  2001:718:1001:2c6::211
radius IN   AAAA  2001:718:1001:2c6::211
client IN   AAAA  2001:718:1001:2c8:76d4:35ff:fe7c:ece
```

### B.3 db.2001.718.1001.2c

```
; Reverzna zona pre domenu raddp.cz.
;
$TTL  86400
@     IN    SOA   raddp.cz. root.raddp.cz. (
                      1       ; Serial
                   604800     ; Refresh
                    86400     ; Retry
                   2419200    ; Expire
                    86400 )   ; Negative Cache TTL
;
@     IN    NS    ns.raddp.cz.
1.1.2.0.0.0.0.0.0.0.0.0.0.0.0.0.6 IN PTR ns.raddp.cz.
e.c.e.0.c.7.e.f.f.f.5.3.4.d.6.7.8 IN PTR client.raddp.cz.
```

# C Výstup príkazu google-authenticator

```
bob@client:~$ google-authenticator --qr-mode=UTF8 -t -d -f --window-size=4 --rat
e-limit=3 --rate-time=30 --issuer=RADDP --label=bob@raddp.cz
https://www.google.com/chart?chs=200x200&chld=M|0&cht=qr&chl=otpauth://totp/bob@
raddp.cz%3Fsecret%3D2CA474VZ6UE5V6C2%26issuer%3DRADDP
```

```
Your new secret key is: 2CA474VZ6UE5V6C2
Your verification code is 588493
Your emergency scratch codes are:
  55015763
  69453101
  66094066
  11279406
  43463963
bob@client:~$
```

Obr. 40: Výstup príkazu `google-authenticator`.

# D Výpisy s logovacích súborov

## D.1 radius.log

```
Fri Apr 22 15:36:13 2016 : Info: rlm_sql (sql): Need 1 more
   connections to reach 10 spares
Fri Apr 22 15:36:13 2016 : Info: rlm_sql (sql): Opening additional
   connection (80), 1 of 30 pending slots used
Fri Apr 22 15:36:13 2016 : Auth: (53) Login OK: [bob] (from client
   private_ipv6_subnet port 29788)
```

## D.2 auth.log

```
Apr 22 15:36:10 eb215-desktop lightdm[31515]: pam_radius_auth: Got
   user name bob
Apr 22 15:36:13 eb215-desktop lightdm[31515]: pam_radius_auth:
   Sending RADIUS request code 1
Apr 22 15:36:13 eb215-desktop lightdm[31515]: pam_radius_auth: Got
   RADIUS response code 2
Apr 22 15:36:13 eb215-desktop lightdm[31515]: pam_radius_auth:
   authentication succeeded
Apr 22 15:36:13 eb215-desktop lightdm(pam_google_authenticator)
   [31515]: debug: start of google_authenticator for bob
Apr 22 15:36:13 eb215-desktop lightdm(pam_google_authenticator)
   [31515]: debug: "/home/bob/.google_authenticator" read
Apr 22 15:36:13 eb215-desktop lightdm(pam_google_authenticator)
   [31515]: debug: shared secret in "/home/bob/.google_authenticator"
    processed
Apr 22 15:36:21 eb215-desktop lightdm(pam_google_authenticator)
   [31515]: debug: no scratch code used from "/home/bob/.
   google_authenticator"
Apr 22 15:36:21 eb215-desktop lightdm(pam_google_authenticator)
   [31515]: debug: "/home/bob/.google_authenticator" written
Apr 22 15:36:21 eb215-desktop lightdm[31515]: pam_fprintd(lightdm):
   Using device /net/reactivated/Fprint/Device/0
Apr 22 15:36:21 eb215-desktop lightdm[31515]: pam_fprintd(lightdm):
   verify_finger_selected Swipe your left index finger across the
   fingerprint reader
Apr 22 15:36:24 eb215-desktop lightdm[31515]: pam_fprintd(lightdm):
   Verify result: verify-match
```